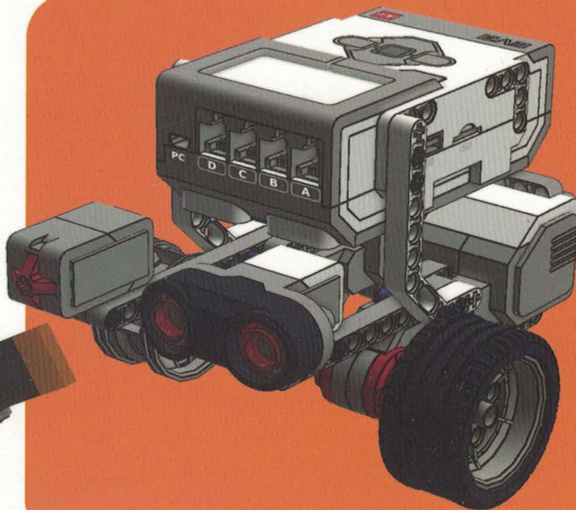
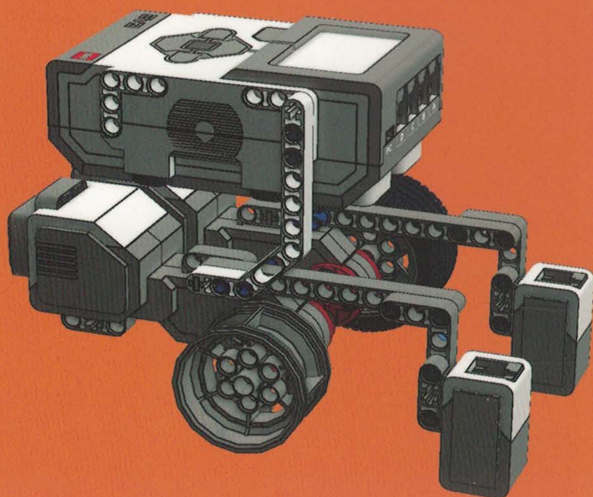
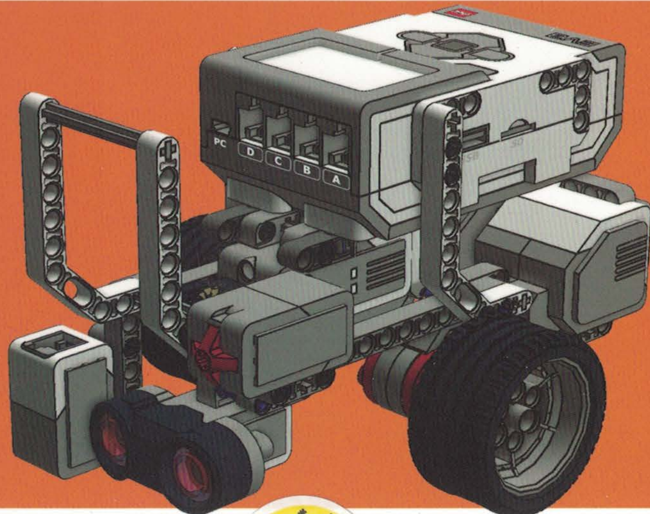
ROBOTC for LEGO EV3 基础编程与实例

码高机器人教育 编著









机械工业出版社
CHINA MACHINE PRESS



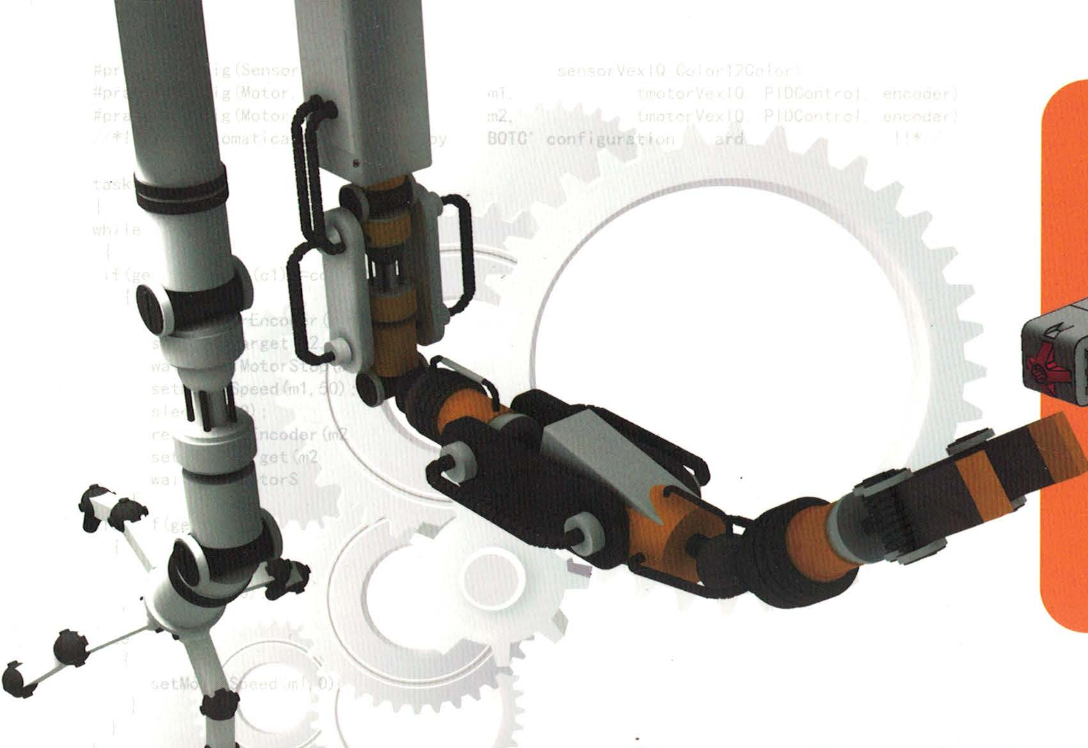
VISIT…

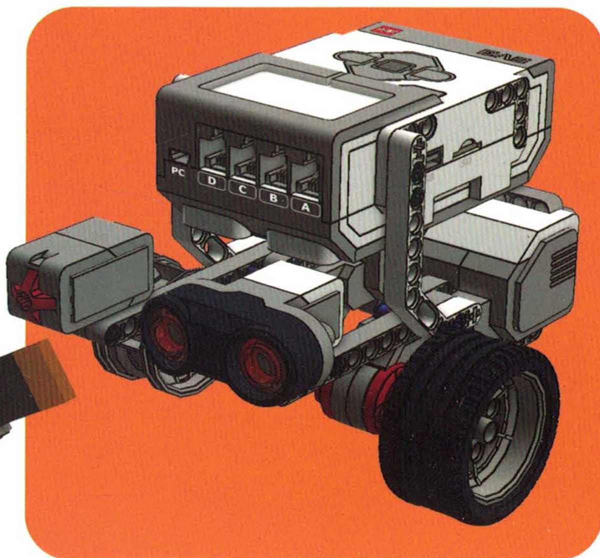
LANZAROTE
Caliente.com

作者简介

码高机器人教育致力于通过提供一站式的机器人教育解决方案，培养孩子的创造力和系统化解决问题的能力。码高机器人教育提供覆盖6~18岁的青少年机器人教育解决方案，机器人爱好者在这里可以完成全部的机器人教育知识学习，熟练掌握机器人的设计、搭建和编程操控，全面提高自身的科学素养，为未来考取理想院校，并成为工程师、程序员，乃至科学家打下坚实的基础。

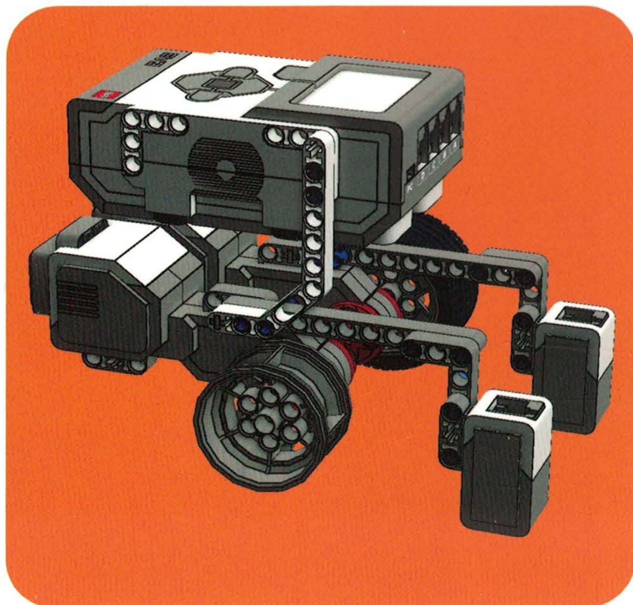
作为一家具备一站式服务能力的机器人教育机构，我们已经打造了常规课程产品线和集训课程产品线。常规课程主要让孩子通过系统化并结合实践的理论学习，掌握扎实的机器人设计、搭建和操控知识。集训产品线则涵盖了国际比赛系列、国际认证系列、国内比赛系列、国内冬夏令营、主题集训等多种产品，通过集中性、团队化、国际化、高难度的训练，让学生从一个操控者变成杰出的机器人驾驭者，能够根据特定的主题规则，设计、搭建和操控自己的机器人，并进行完美的任务挑战，与国际和国内的机器人驾驭高手同场竞技，实现创造力和系统化解决问题能力的全面提升。

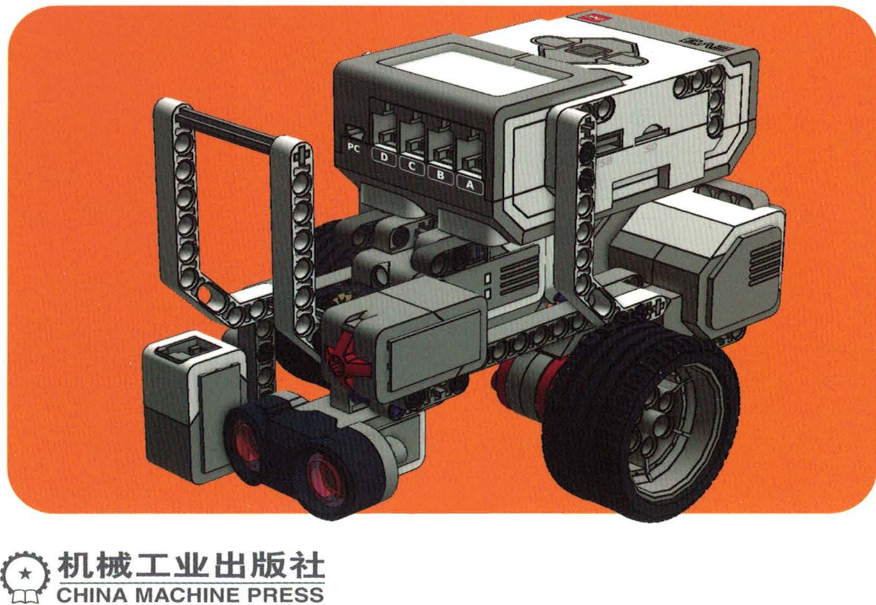




ROBOTC for LEGO EV3 基础编程与实例

码高机器人教育 编著





机械工业出版社
CHINA MACHINE PRESS

ROBOTC 是由卡耐基梅隆大学机器人学院开发的基于 C 语言的机器人编程工具，它采用标准 C 语言，拥有丰富的程序编写功能和独特创新的调试功能，支持 VEX IQ、VEX EDR、LEGO MINDSTORMS 等。

本书以 LEGO MINDSTORMS 作为机器人平台，通过全方位的编程实例讲解，深入介绍 ROBOTS 的编程方法。读者可以发挥自己的想象力，动手搭建心目中的乐高任务机器人，同时又可以学习到 C 语言的基础编程知识，为将来的学习和工作打下良好的基础。

参与本书编写的人员有董刚红、叶昌青、李晶、张金宝、王好强。

图书在版编目(CIP)数据

ROBOTC for LEGO EV3 基础编程与实例 / 码高机器人教育编著. —北京：机械工业出版社，2018. 5

ISBN 978-7-111-60056-5

Ⅰ. ①R… Ⅱ. ①码… Ⅲ. ①机器人-程序设计 Ⅳ. ①TP242

中国版本图书馆 CIP 数据核字(2018)第 102255 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：杨 源 责任编辑：杨 源
责任校对：秦洪喜 责任印制：李 昂
北京联兴盛业印刷股份有限公司印刷
2018 年 6 月第 1 版第 1 次印刷
215mm×225mm · 4. 6 印张 · 200 千字
0001—3500 册
标准书号：ISBN 978-7-111-60056-5
定价：49. 80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务 网络服务
服务咨询热线：010-88361066 机工官网：www. cmpbook. com
读者购书热线：010-68326294 机工官博：weibo. com/cmp1952
010-88379203 金书网：www. golden-book. com
封面无防伪标均为盗版 教育服务网：www. cmpedu. com

目 录

第 1 章 ROBOTC for LEGO 软件介绍 …… 1

1. 1 ROBOTC 简介 …… 1

1. 2 ROBOTC 安装 …… 2

1. 3 ROBOTC 界面 …… 10

1. 4 ROBOTC 虚拟世界 …… 31

第 2 章 编程讲解与实例 …… 45

2. 1 编程基础讲解 …… 45

2. 2 迷宫挑战 …… 52

2. 3 电动机控制 …… 55

2. 4 触动传感器 …… 57

2. 5 超声波传感器 …… 60

2. 6 陀螺仪传感器 …… 61

2. 7 颜色传感器 …… 63

2. 8 巡线 …… 69

2. 9 显示与 LED …… 71

2. 10 变量 …… 73
2. 11 声音 …… 77
2. 12 函数 …… 79
2. 13 switch. . . case …… 82
2. 14 线程 …… 85
2. 15 随机数 …… 86

第 1 章　ROBOTC for LEGO 软件介绍

1.1　ROBOTC 简介

ROBOT C 是由卡耐基梅隆大学机器人学院开发的基于 C 语言的机器人编程工具，它采用标准 C 语言，拥有丰富的程序编写功能和独特创新的调试功能，支持 VEX IQ、VEX EDR、LEGO MINDSTORMS 等。本书以 LEGO MINDSTORMS 作为机器人平台，通过一些实例，深入介绍了 ROBOTC 的编程和应用。

机器人是一个理想的创新教育工具，在课程中，学生可以学到结构搭建和程序设计。近年来，国内对机器人教育的重视程度与日俱增。从小学到中学，从中学到大学，各种类型的机器人竞赛应有尽有。机器人竞赛推动了机器人课程的学习。由学习到参与竞赛，再由竞赛到进一步学习，形成一个良性循环。学生在机器人竞赛的过程中能学习团队合作、全力以赴的精神。这些能力对学生的发展与成长具有非常深远的影响。

ROBOTC 与市面上众多的机器人开发环境相比，主要有以下几方面优点：

（1）功能齐全

ROBOTC 拥有编写和调试文本程序的所有功能和成熟的机器人程序设计调试工具。它具备完整的编辑菜单、C 语言数组边界检查等功能。在调试方面给予用户最大限度的实时调试功能。对 VEX 的各项功能提供了良好的支持。

（2）体积小巧

界面简洁朴素，只有简单而且必要的，但完全足够使用的功能菜单，非常节省资源。对计算机配置的要求非常低。

（3）拓展性好

ROBOTC 语言支持 LEGO MINDSTORMS（包括 LEGO 拓展套件 TETRIX 和 MATRIX）、VEX EDR 以及 VEX IQ 等多种机器人平台。

（4）实用价值高

C 语言是国内大部分理工科院校学生学习程序设计的入门语言，掌握 ROBOTC 语言就能为以后学习 C 语言打下基础。

（5）便捷直观

在下载完程序后，自动出现的调试窗口可以迅速直观地向用户展示程序运行的内部情况。

ROBOTC for VEX Robotics 4. X 允许用户使用一个全新的图形化编程界面或者使用标准的 C 语言编程，在同一软件中编缉 VEX IQ 机器人。

对于初始用户，新的 ROBOTC 图形化编程界面，让用户可以通过使用诸如“向前”“向右转”“线轨道”“街机控制”之类的命令快速启动和运行机器人。用户可以定制机器人配置，并使用任何机器人配置在图形化编程界面中进行编程。

在 ROBOTC 中，用户可以用专业工程师和计算机科学家所使用的标准 C 语言编程，掌握这项技能便可以在未来的科技时代与社会更好地接轨。

ROBOTC Robotics 4. X 包括专门为 VEX IQ 设计的 100 多个新的命令和 200 多个示例程序，让用户了解如何使自已的机器人移动和感应。

1. 2　ROBOTC 安装

首先要先下载 ROBOTC 软件安装包。

（1）双击安装包开始安装，等待一分钟左右完成安装准备，如图 1-1 所示。

（2）进入安装向导，然后单击 Next 按钮进入下一步，如图 1-2 所示。

（3）在用户协议中选“I accept the terms in the license agreement”，然后单击“Next”按钮进入下一步，如图 1-3 所示。

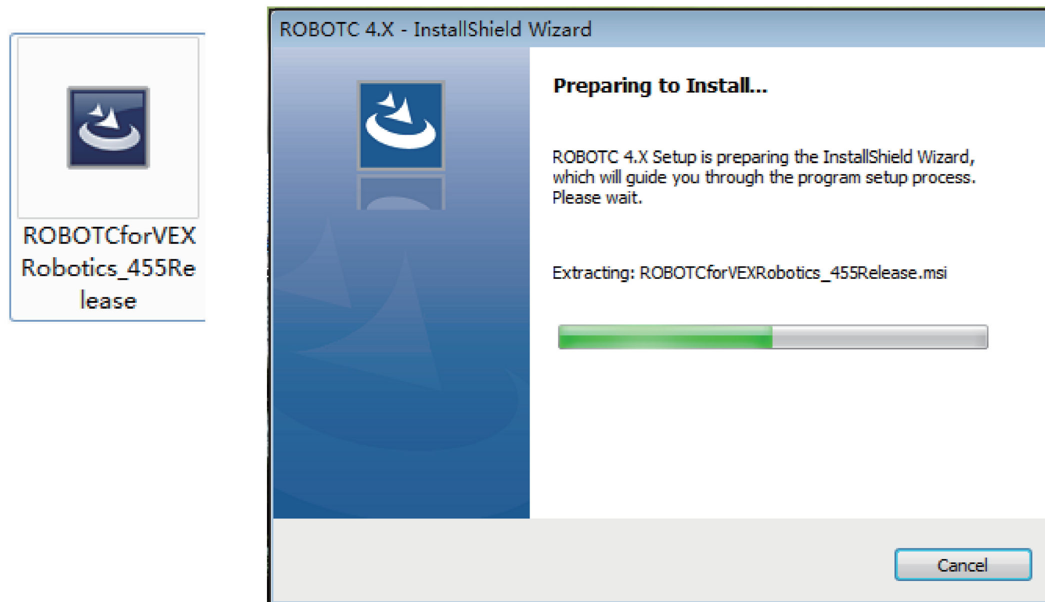

ROBOTCforVEX Robotics_455Release
ROBOTC 4.X - InstallShield Wizard
Preparing to Install...
ROBOTC 4.X Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.
Extracting: ROBOTCforVEXRobotics_455Release.msi
Cancel

图 1-1

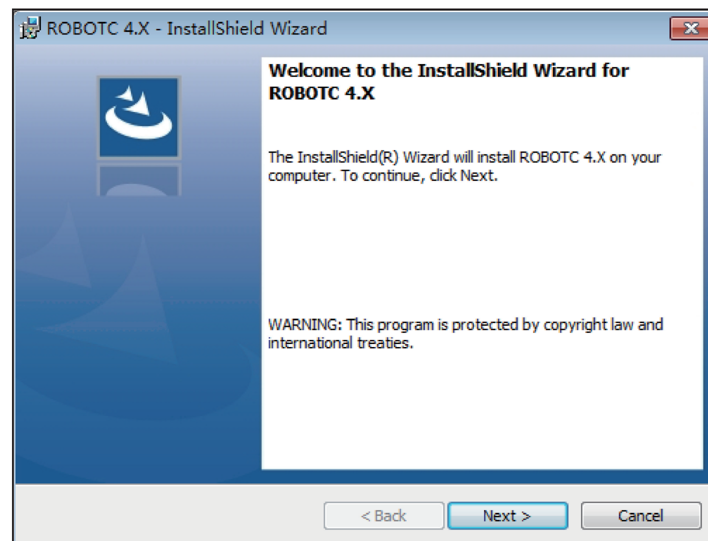

ROBOTC 4.X - InstallShield Wizard
Welcome to the InstallShield Wizard for ROBOTC 4.X
The InstallShield(R) Wizard will install ROBOTC 4.X on your computer. To continue, click Next.
WARNING: This program is protected by copyright law and international treaties.
< Back
Next >
Cancel

图 1-2

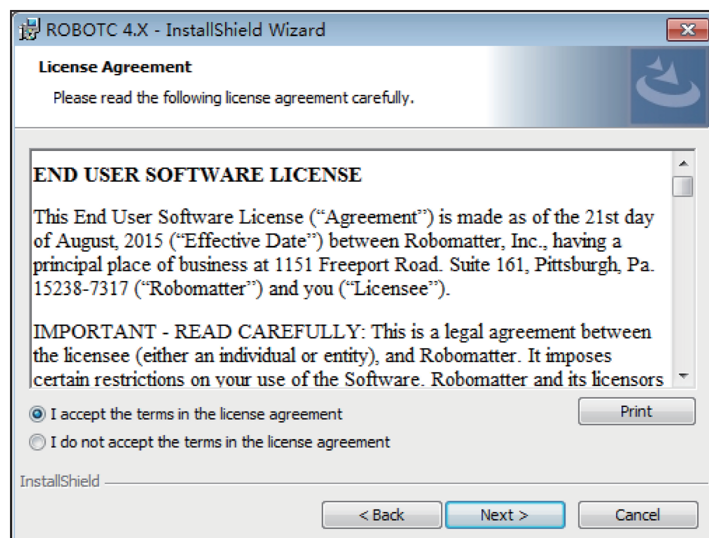


图 1-3

（4）因为 ROBOTC 软件的特殊性，在安装时不需修改安装路径，直接安装到 C:盘即可，如图 1-4 所示。

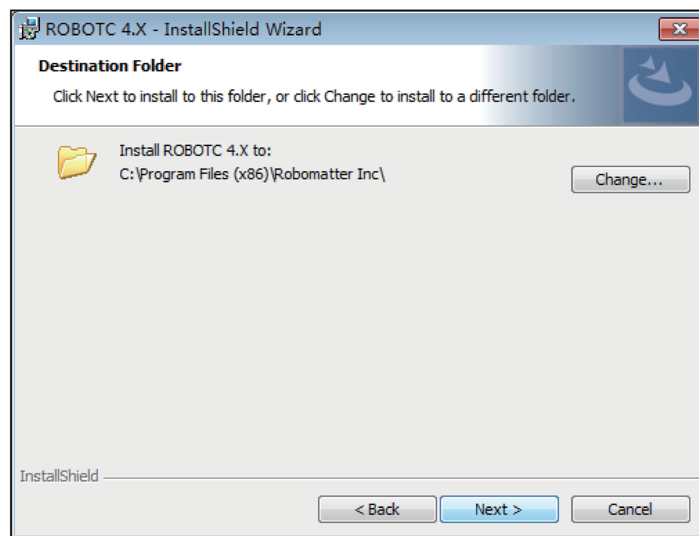


图 1-4

（5）安装方式有两种，一种是完全安装，一种是定制安装，选择完全版“Complete”进行安装，然后单击 Next 按钮进入下一步，如图 1-5 所示。

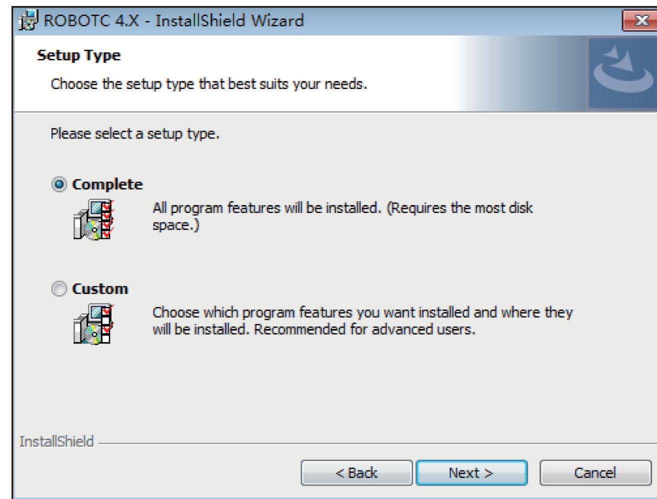


图 1-5

（6）单击 Install 按钮进行安装，如图 1-6 所示。

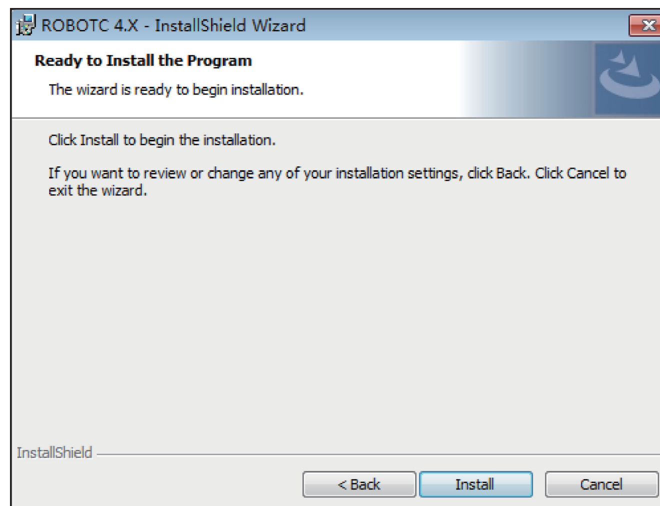


图 1-6

（7）等待软件自动安装即可，如图 1-7 所示。

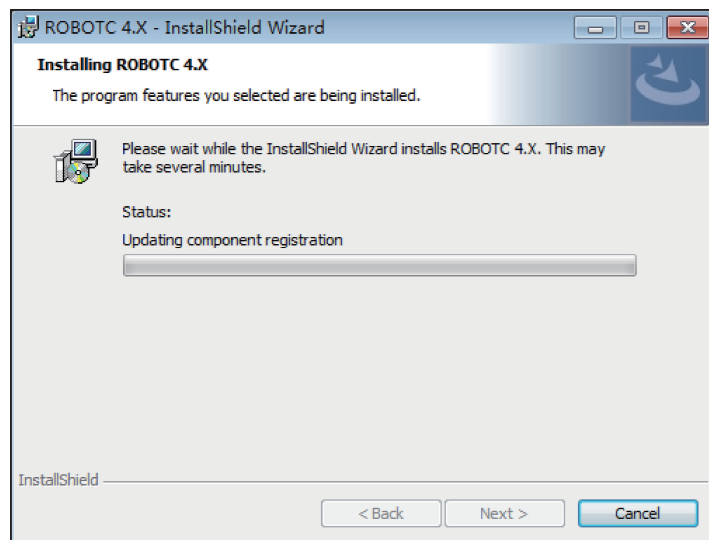


图 1-7

在安装过程中会出现 C:盘窗口，不必理会，静候几分钟即可，如图 1-8、图 1-9 所示。

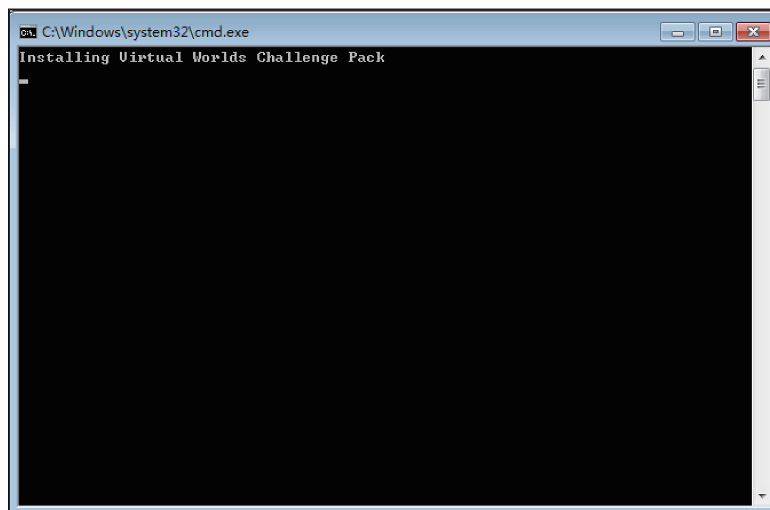


图 1-8

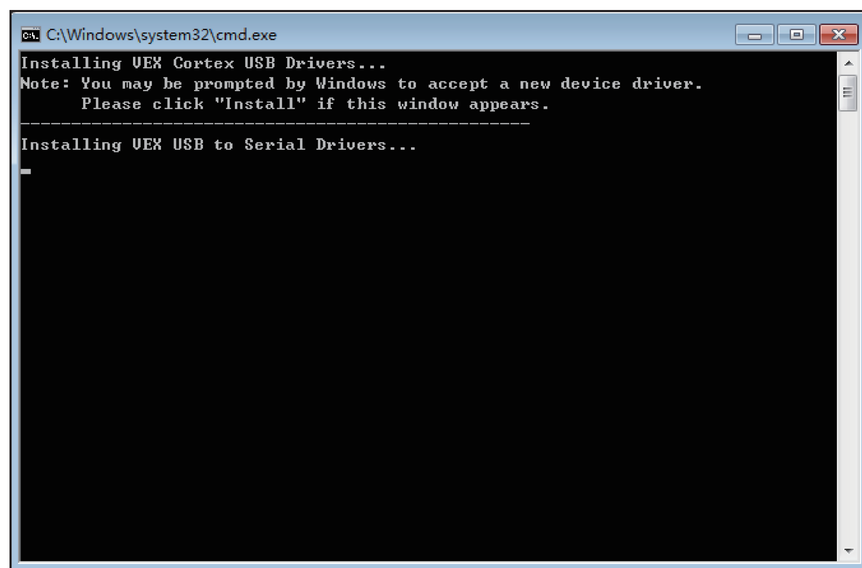


图 1-9

（8）单击 Finish 按钮完成 ROBOTC 软件的全部安装，如图 1-10 所示。

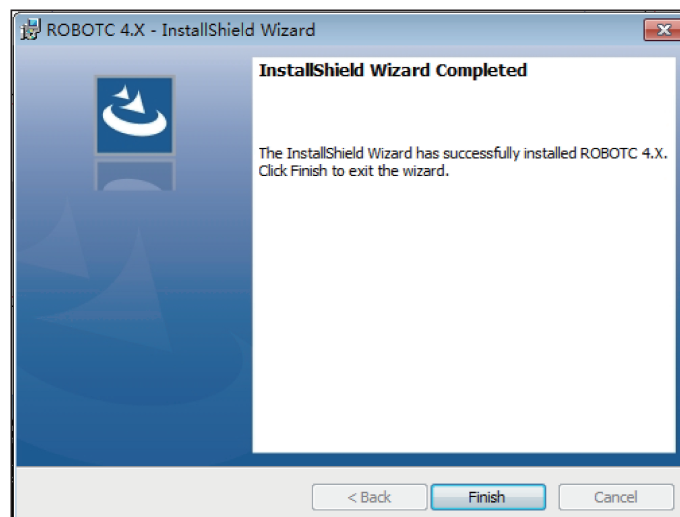


图 1-10

（9）这时桌面会生成两个快捷方式，第一个 Graphical ROBOTC for VEX Robotics 4. X 为图形化编程，第二个 ROBOTC for VEX Robotics 4. X 为 C 语言编程，本书将着重介绍第二个，也是编程常用的 ROBOTC，如图 1-11 所示。

（10）双击快捷方式，当出现安全警报窗口时，单击“是”按钮即可，如图 1-12、图 1-13 所示。

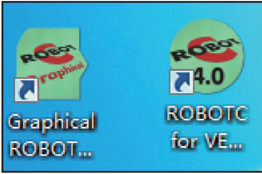


图 1-11

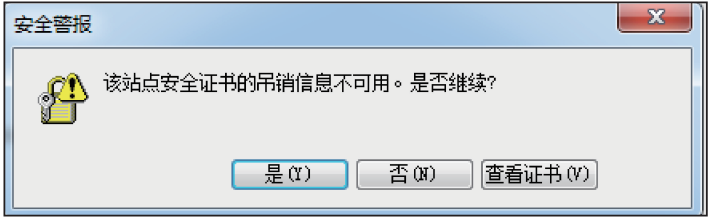


图 1-12



图 1-13

（11）进入软件之后，先选择 Help/Add License 命令，进行证书选择，如图 1-14 所示。

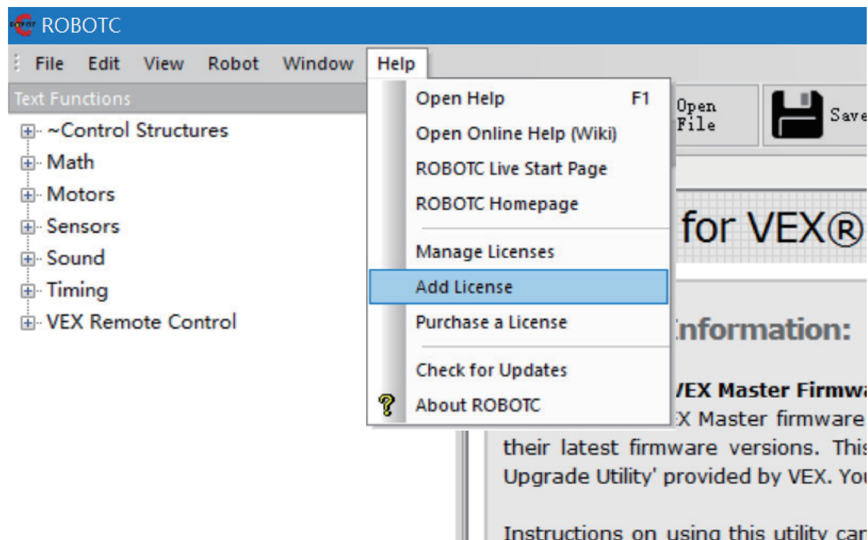


图 1-14

（12）在证书选项中选择第一项“ROBOTC for LEGO MINDSTORMS 4. X”进行激活，如图 1-15 所示。



图 1-15

（13）图 1-15 中的证书意思分别是 LEGO 的 ROBOTC、LEGO 的虚拟世界和 VEX 的虚拟世界，注意，每个证书可以试用 10 天，如果要长期使用，需要到官方网站进行购买。在这里可以单击 Start Trail 按钮进行试用。如果是正版激活，就输入证书 id 和 password，然后单击 Activate Online 按钮。

这时桌面多了几个图标，如图 1-16 所示。



图 1-16

黄色图标即为 LEGO 的 ROBOTC 和虚拟世界，在这里使用 ROBOTC for LEGO MINDSTORMS，进入软件。

1.3 ROBOTC 界面

下面来了解 ROBOTC 界面的基本情况。

打开 ROBOTC 软件时，显示画面如下图所示，单击 New File 按钮，即可开启我们的编程之旅。首先要对 ROBOTC 软件的界面进行一个全方位介绍，如图 1-17 所示。

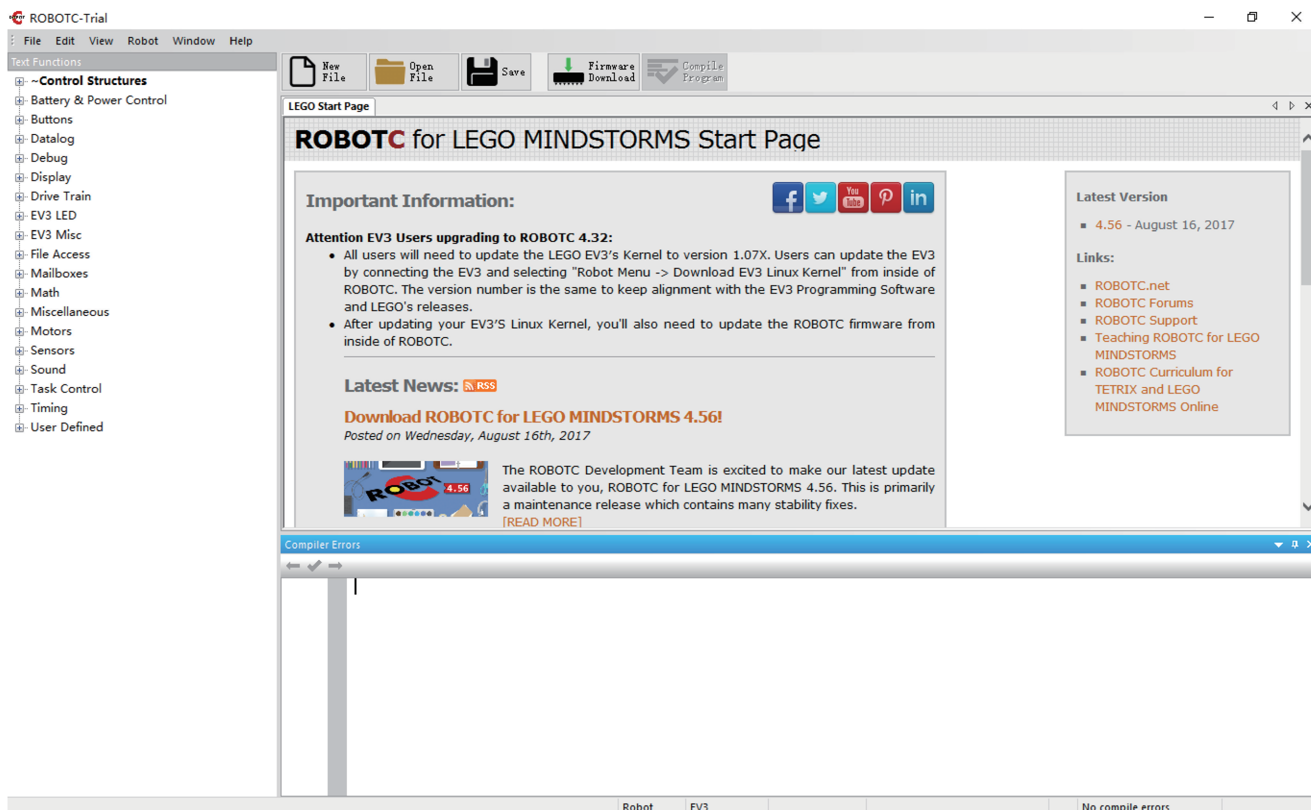


图 1-17

菜单栏：（常用菜单）包括文件、编辑、视图、机器人、窗口、帮助 6 个菜单。

工具栏：（常用工具）包括新建、打开、保存、固定格式、电动机和传感器设置、固件下载、编译程序、下载到机器人 8 个工具。

函数库：在编程过程中有时候会应用到各种函数，这些函数均可以从函数库中调出。

编程区：所有 ROBOTC 编程语言都将在此区域编写。

编译区：当程序出现问题或错误时，在编译区会有显示。在主控器运行时，可以显示超声波传感器、颜色传感器、陀螺仪传感器等运行的状态信息（超声波探测障碍物的距离、颜色传感器探测的灰度值、陀螺仪旋转的角度等），如图 1-18 所示。

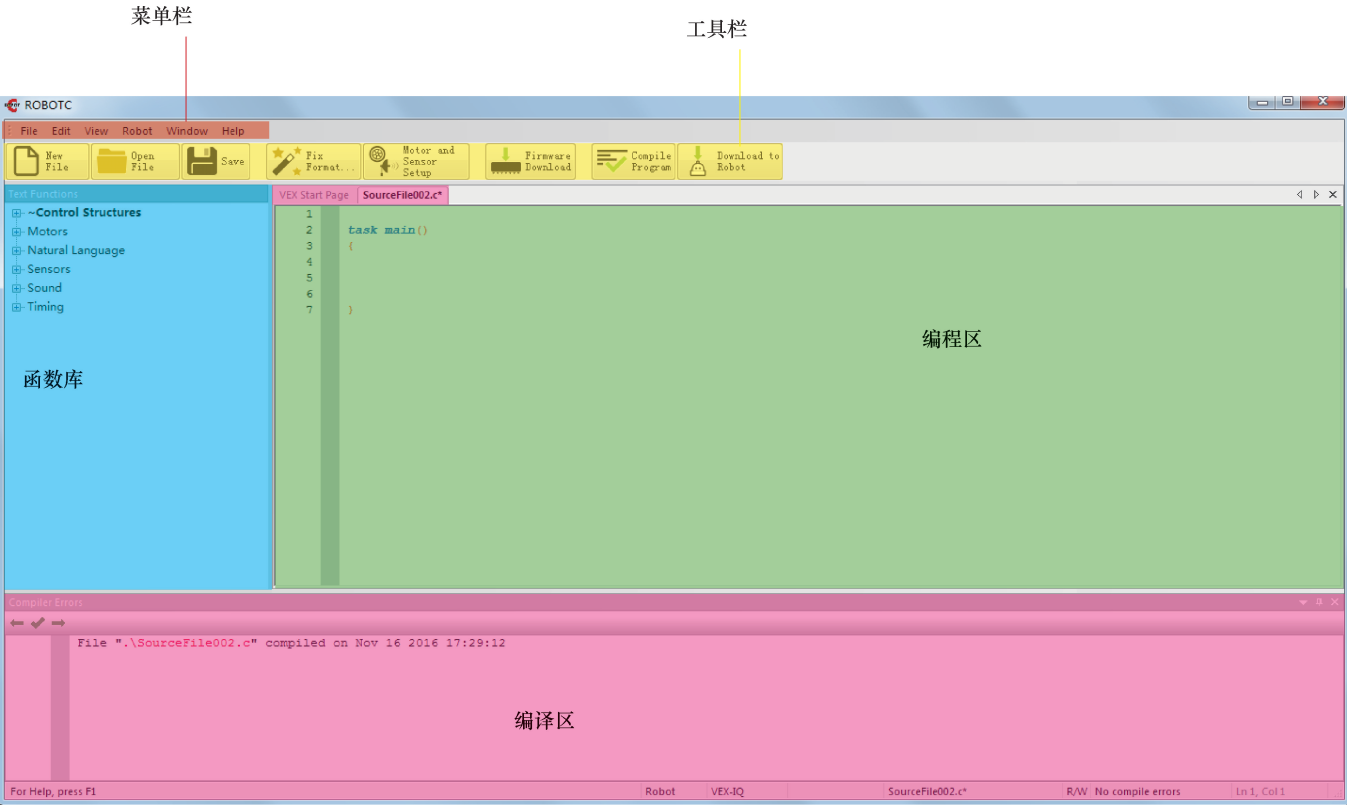


图 1-18

1.3.1 菜单栏

菜单栏如图 1-19 所示。

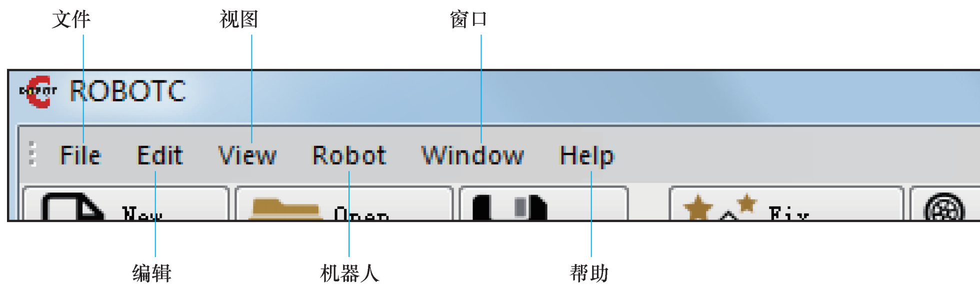


图 1-19

1. 文件界面

文件界面如图 1-20 所示。

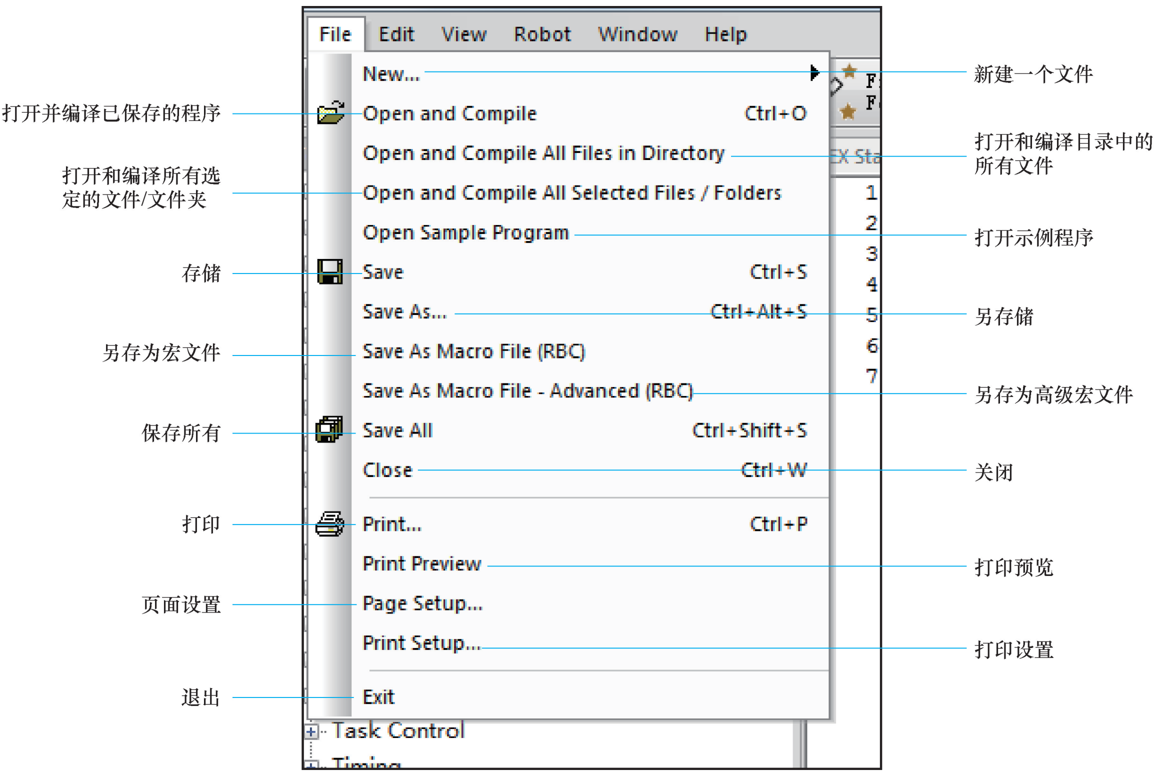


图 1-20

2. 编辑界面

编辑界面如图 1-21 所示。

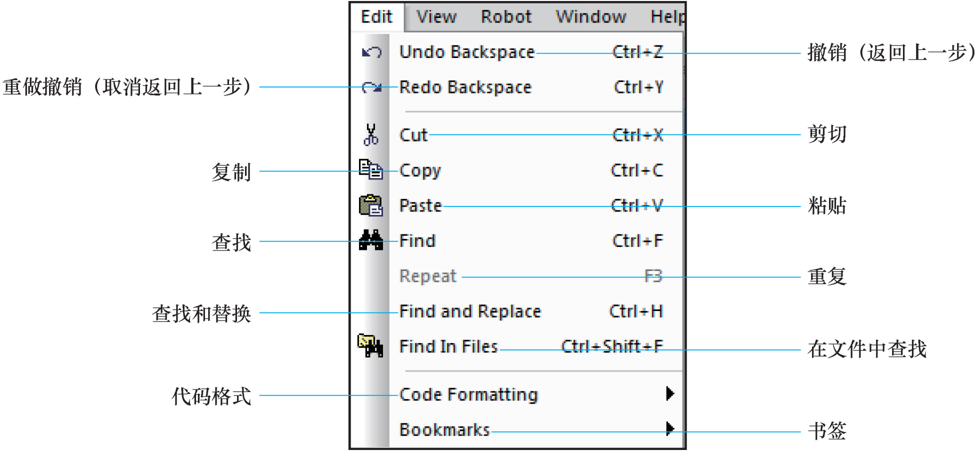


图 1-21

（1）编程 – 代码格式

编程 – 代码格式，这一功能不常使用，所以不多做介绍，如图 1-22 所示。

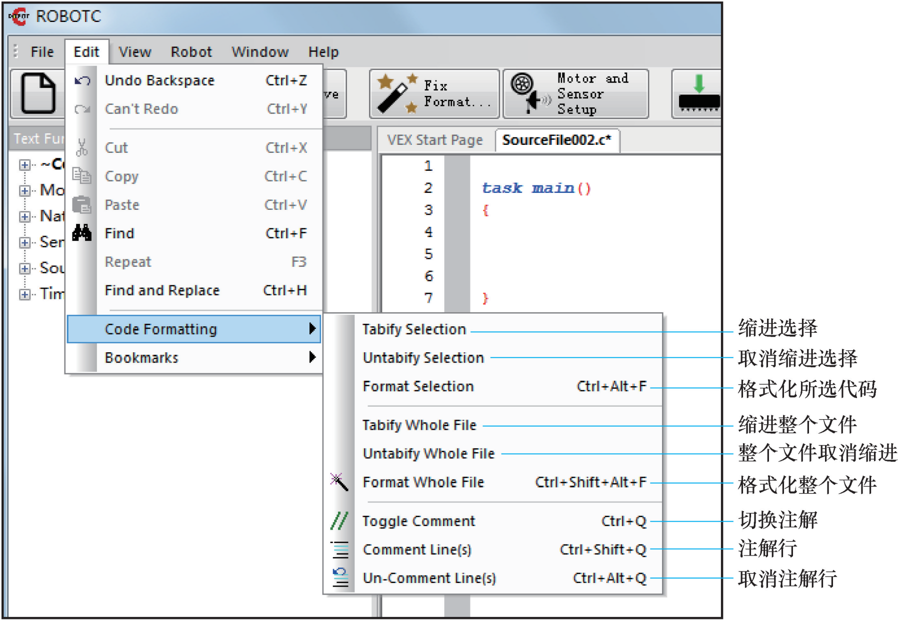


图 1-22

（2）编程 – 书签

可以使用这个功能为编程添加书签，方便地找到一些重要的信息，如图 1-23 所示。

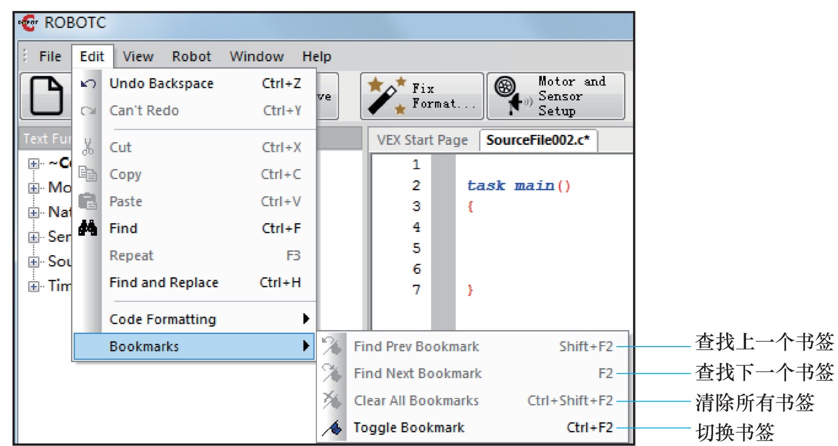


图 1-23

3. 视图界面

视图界面如图 1-24 所示。



图 1-24

（1）视图－系统文件

系统文件（头文件）：在 C 语言家族中，头文件被大量使用。一般而言，每个 C＋＋/C 程序通常由头文件和定义文件组成。头文件作为一种包含功能函数、数据接口声明的载体文件，主要用于保存程序的声明，而定义文件用于保存程序的实现。这里只做了解即可，如图 1-25 所示。

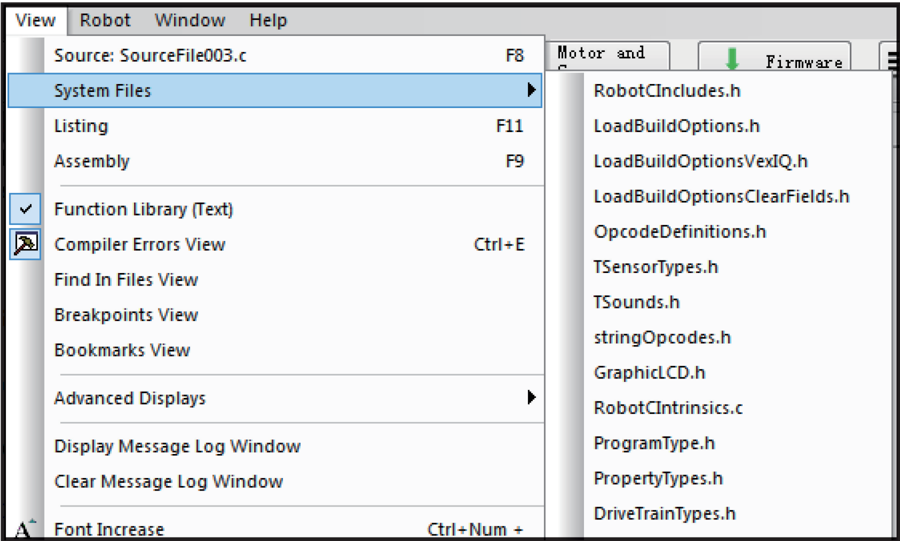


图 1-25

（2）视图－高级显示

高级显示如图 1-26 所示。

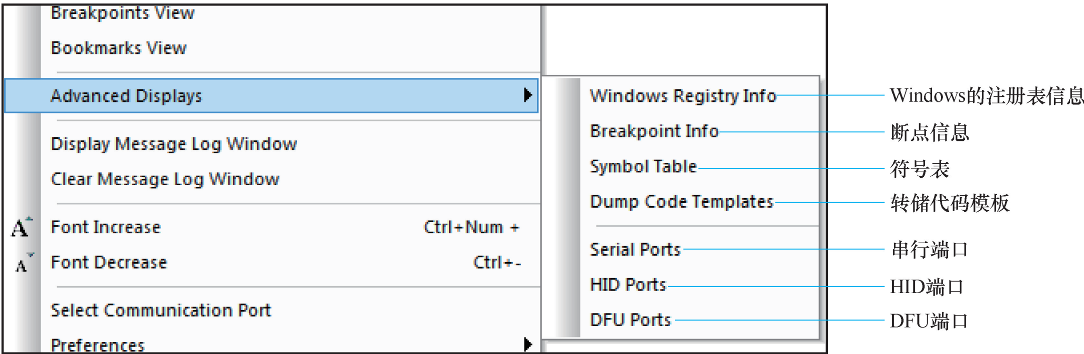


图 1-26

(3) 视图 – 首选项

首选项如图 1-27、图 1-28 所示。

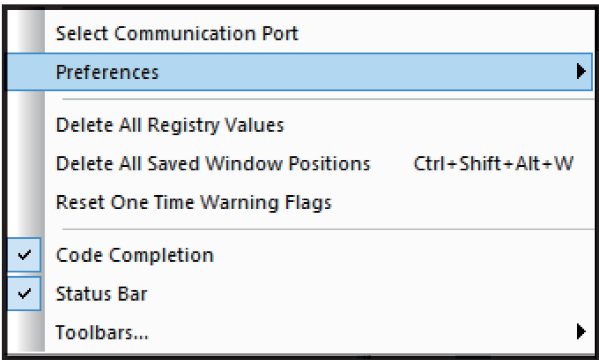


图 1-27

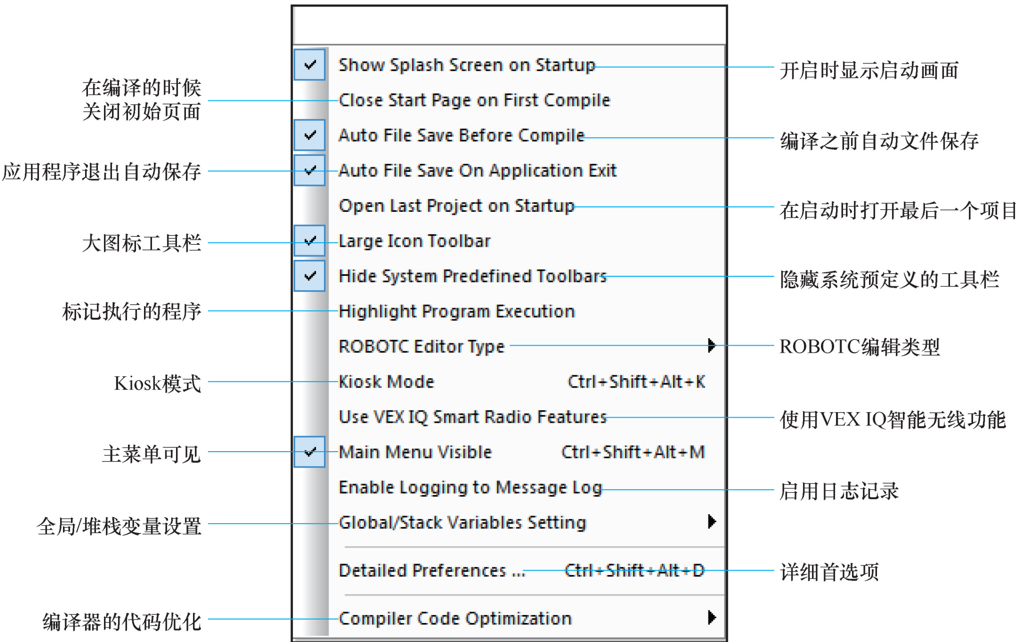


图 1-28

（4）首选项 – ROBOTC 编辑类型

ROBOTC 编辑类型如图 1-29 所示。

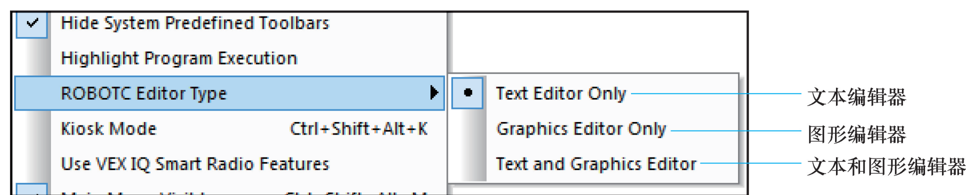


图 1-29

（5）首选项 – 全局/堆栈变量设置

全局/堆栈变量设置如图 1-30 所示。

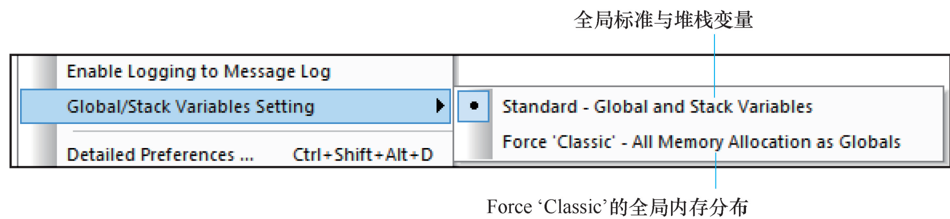


图 1-30

（6）首选项 – 编译器的代码优化

编译器的代码优化如图 1-31 所示。

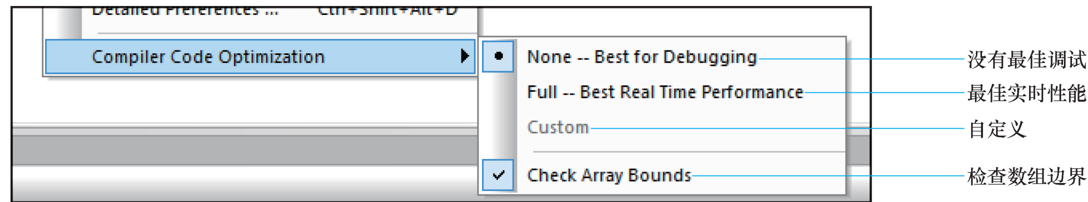


图 1-31

（7）视图 – 工具栏

工具栏如图 1-32 所示。

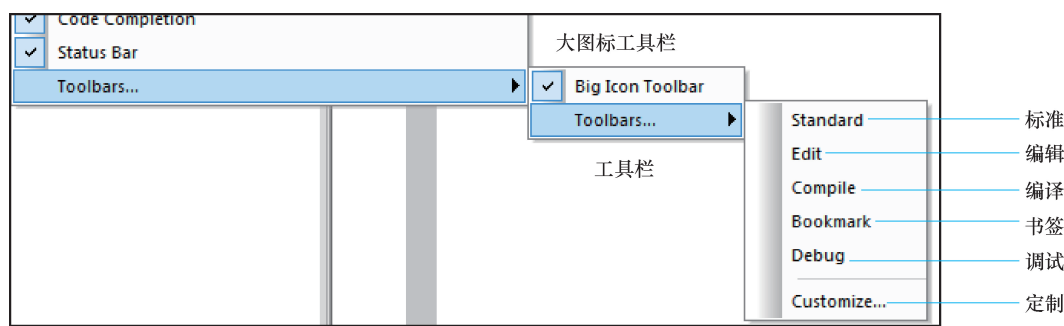


图　1-32

4. ROBOTC 界面

ROBOTC 界面菜单如图 1-33 所示。

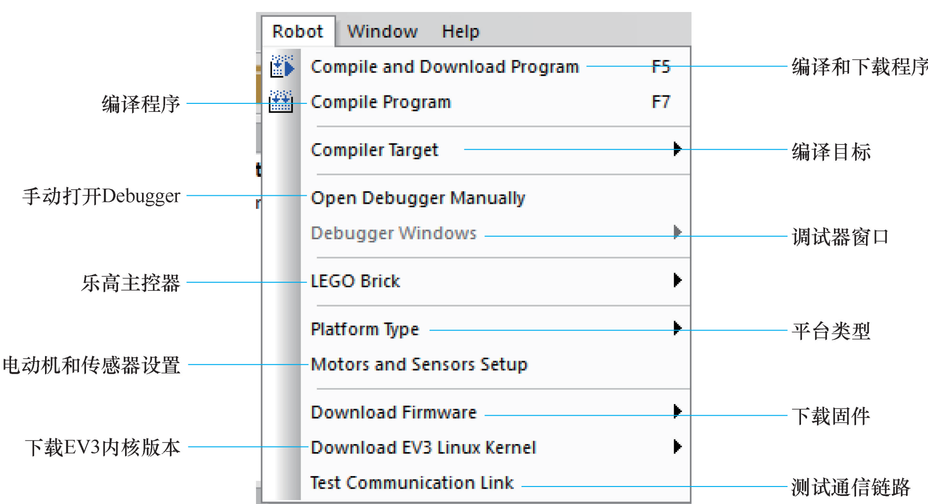


图　1-33

（1）机器人 – 编译器的目标

编译器的目标如图 1-34 所示。

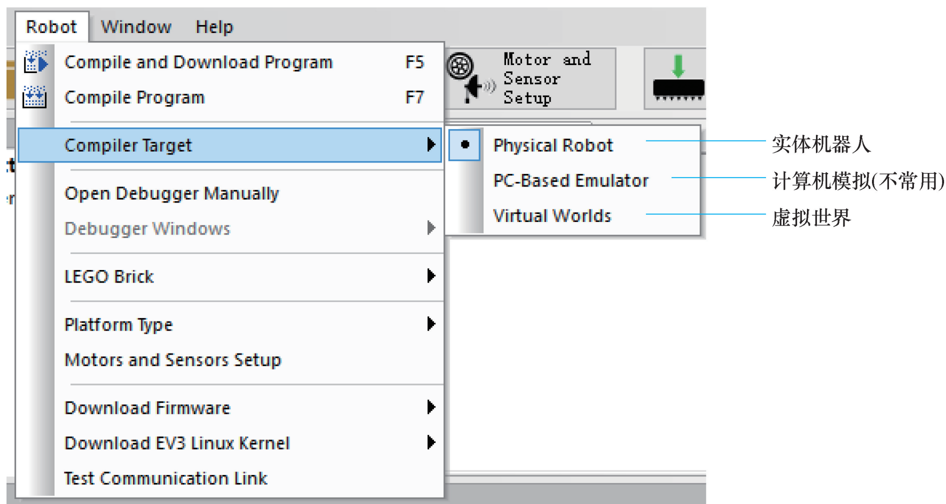


图 1-34

（2）机器人乐高主控器

乐高主控器如图 1-35 所示。

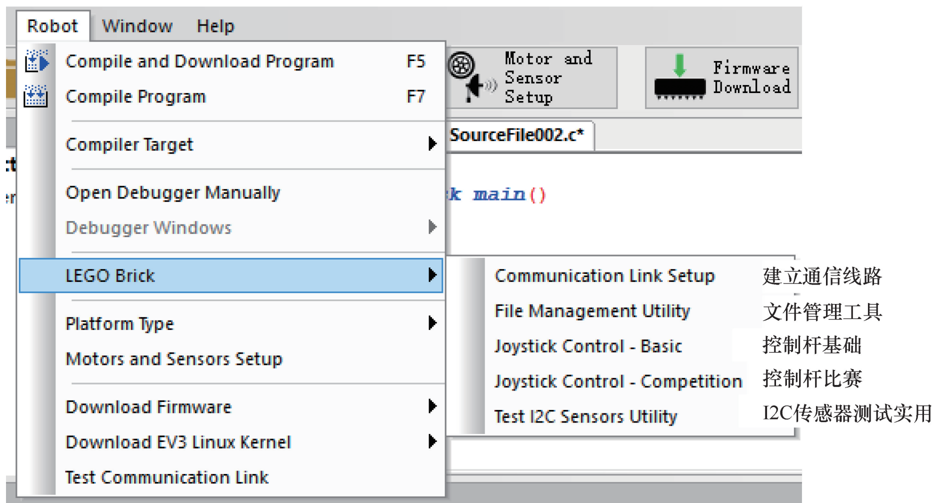


图 1-35

（3）机器人 – 平台类型

平台类型如图 1-36 所示。

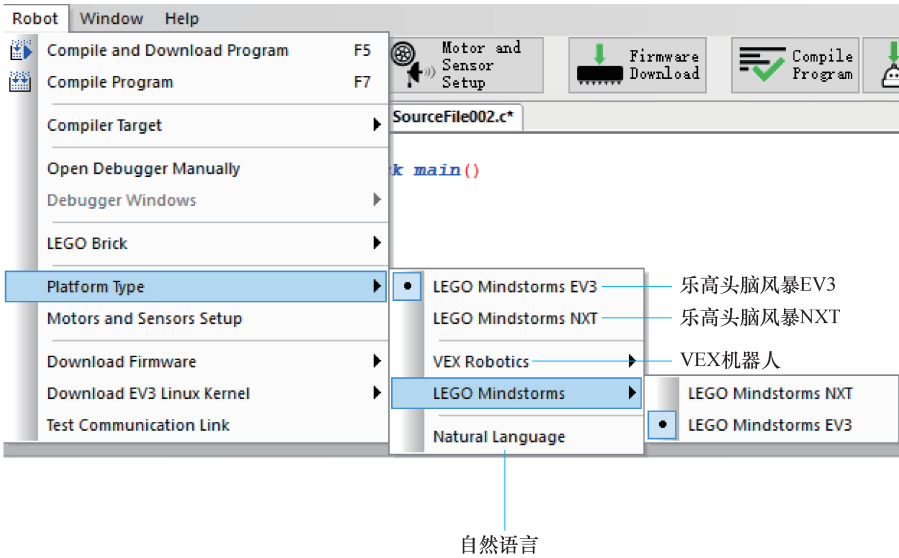


图 1-36

（4）机器人 – 下载内核版本

计算机连接到 EV3 之后，要为 EV3 下载新固件，否则可能下载不了程序，选择下载 EV3 内核版本中的第一个选项，准备文件，根据提示框进行操作，如图 1-37 所示。

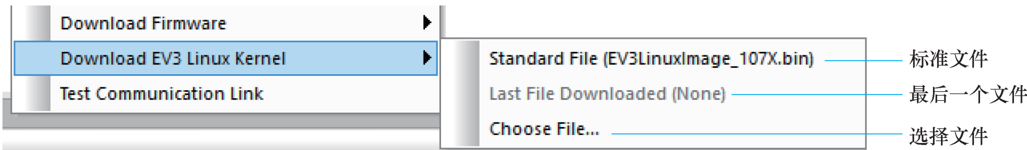


图 1-37

这个时候 EV3 上会显示 updating，如果报错，不用理会，进行重复操作，多刷新几次，有时候会出现超时的情况，等待进度条完成，如图 1-38 所示。

升级成功，等待 EV3 主控器重新启动，重启完成之后，就会显示 OK。

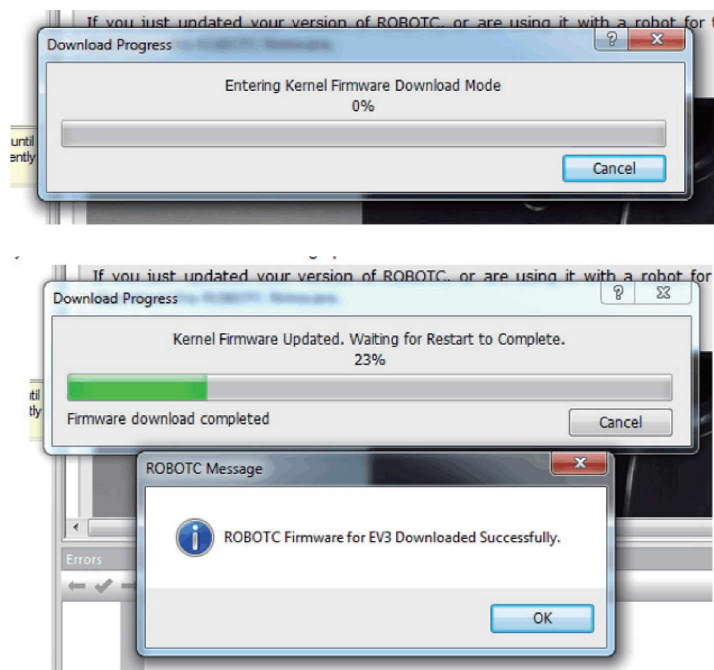


图 1-38

可以将 ROBOTC 用 USB、蓝牙连接 EV3，也可以通过蓝牙下载程序（过程比较慢，推荐使用数据线下载）。

有时候在升级 EV3 固件之后，名字被重置为“EV3-3”，且不能修改。此时只能打开 lEGO MINDSTORMS，连接 EV3，在下图提示处修改名字，新的 ROBOTC 固件仍然兼容乐高中级班的图形编程如图 1-39 所示。



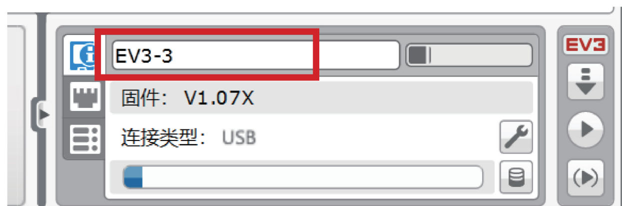


图 1-39

5. 窗口界面

窗口界面如图 1-40 所示。

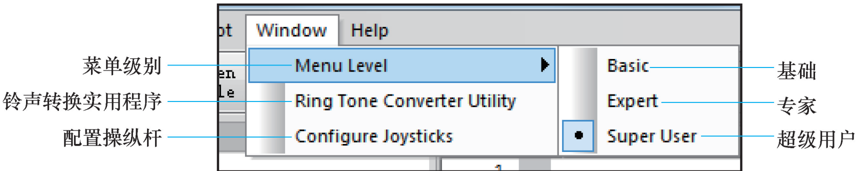


图 1-40

6. 帮助界面

帮助界面如图 1-41 所示。

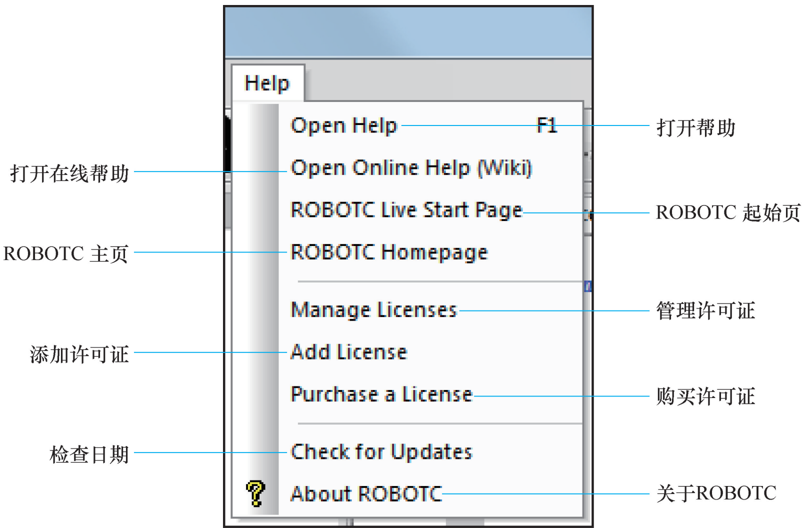


图 1-41

单击 Open Help 命令打开帮助界面，首先要选择语言，可选择 English，然后单击 OK 按钮，进入下一步，如图 1-42 所示。

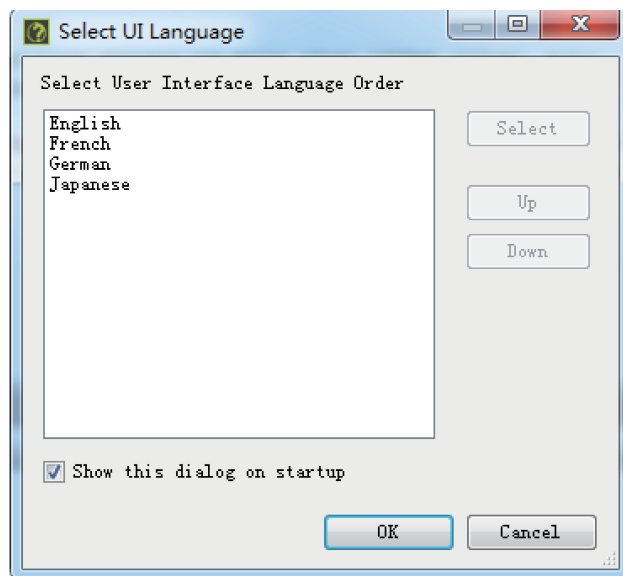


图 1-42

出现如下界面，等待几秒钟便可以进入帮助窗口中，如图 1-43 所示。



图 1-43

在帮助窗口可以根据自己的需求查看帮助内容。也可以单击工具栏下的搜索按钮，如图 1-44 所示。

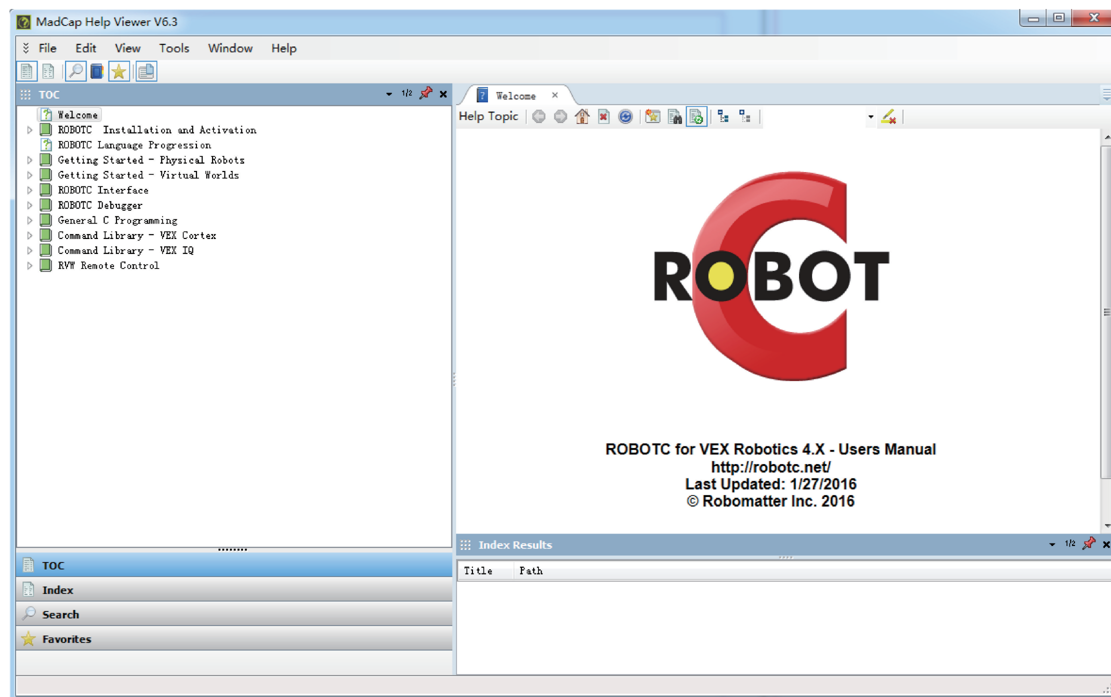


图 1-44

在搜索栏中输入要搜索的信息，输入完毕之后单击 Search 按钮进行搜索，如图 1-45 所示。

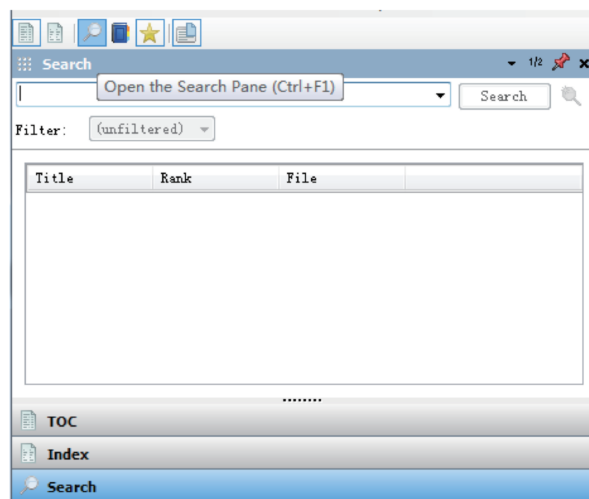


图 1-45

这时在搜索结果中将会显示对目标搜索的所有结果，然后选择所需的信息，窗口右侧会显示该信息所有详解内容，如图 1-46 所示。

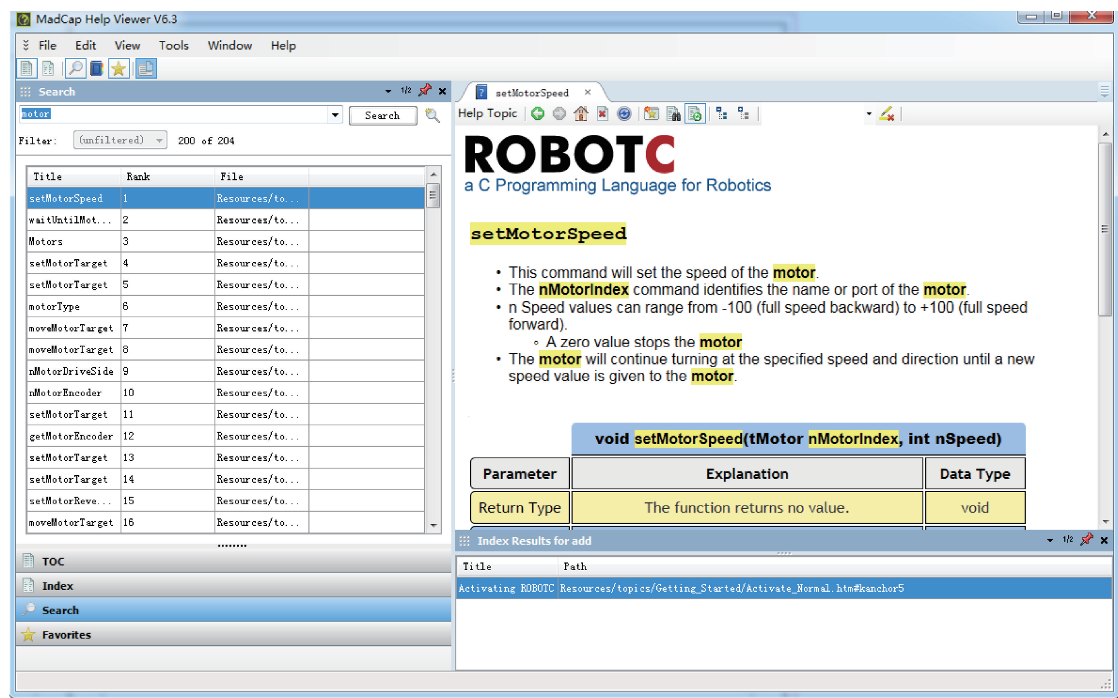


图 1-46

1.3.2 工具栏

新建文件、打开文件、存储，与文件菜单中的功能一样，放在工具栏中是为了更方便操作，所以直接从电动机和传感器设置开始介绍，如图 1-47 所示。

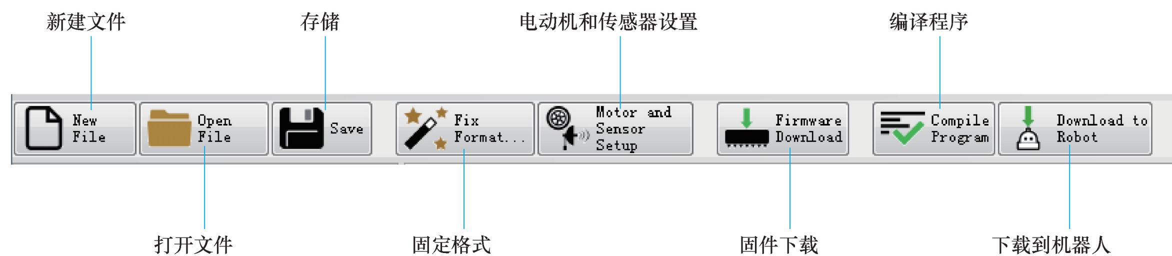


图 1-47

一般情况下在标准模型和数据采样中设置都不用修改，默认即可，然后直接进行电动机与传感器的设置，如图 1-48 ~ 图 1-50 所示。

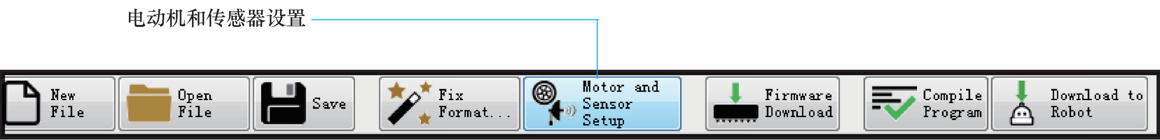
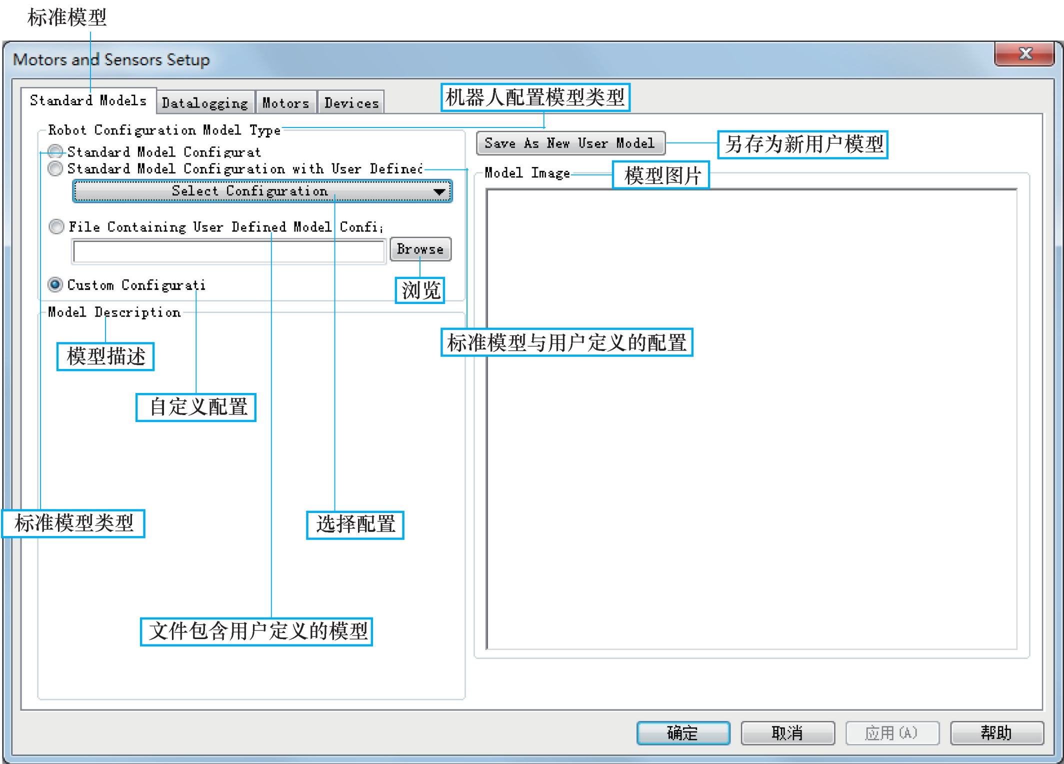


图 1-48



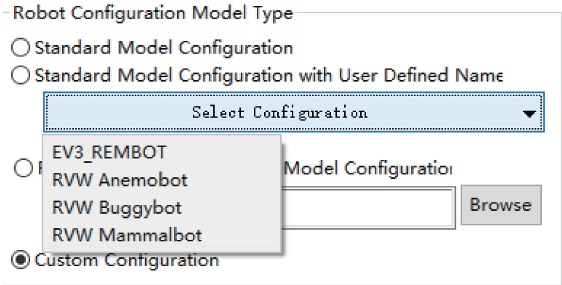


图 1-49

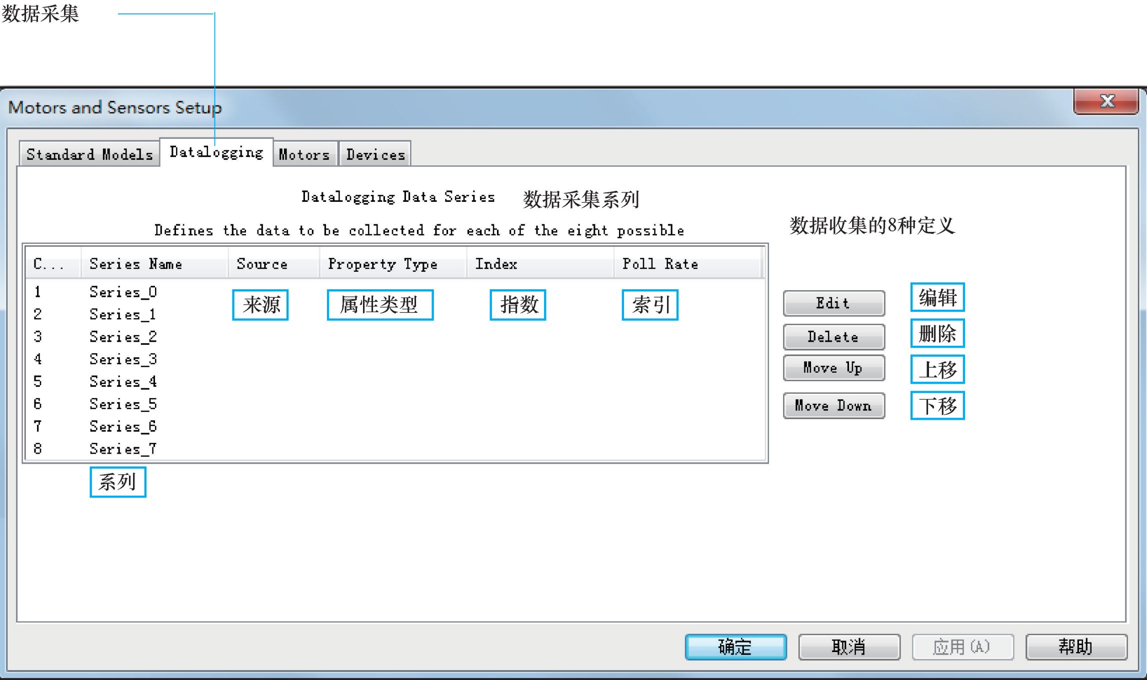


图 1-50

在这里可以根据 EV3 机器人安装的电动机端口，对电动机进行设置命名，这是在编程之前必要的一步，如图 1-51 ~ 图 1-53 所示。

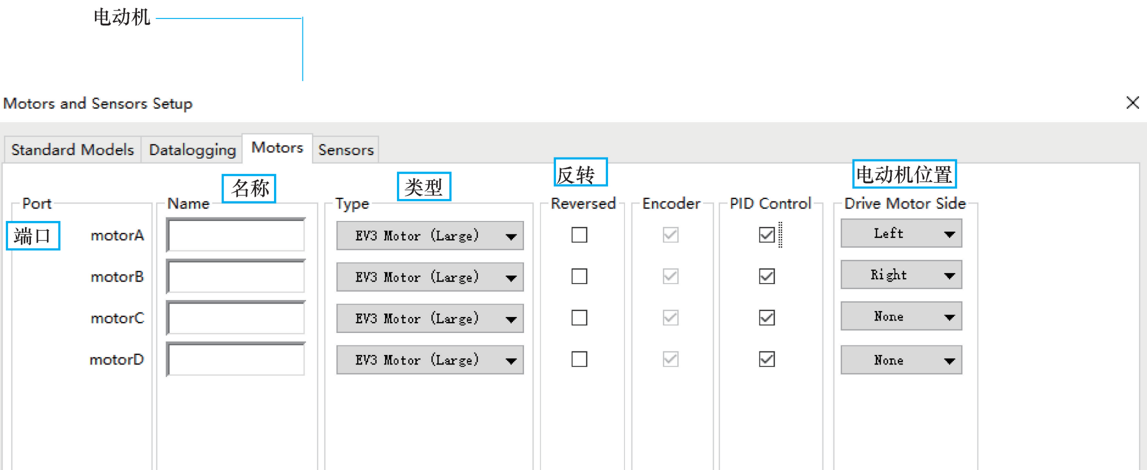


图 1-51

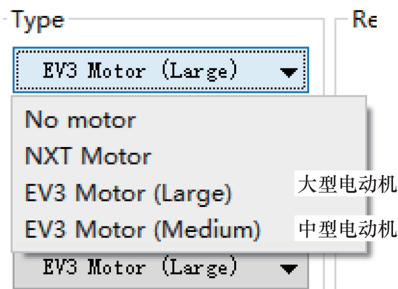


图 1-52

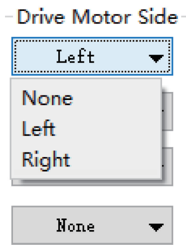


图 1-53

跟电动机设置相同，也可以在这里对 EV3 的传感器相对应的端口进行设置如图 1-54 ~ 图 1-56 所示。

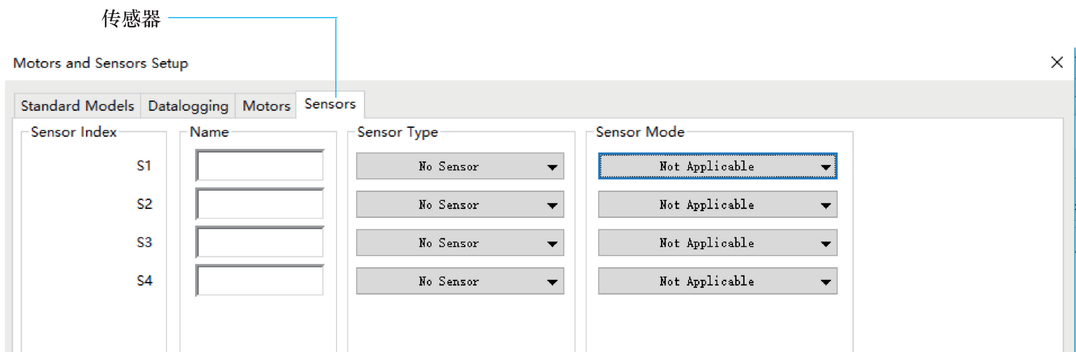


图 1-54

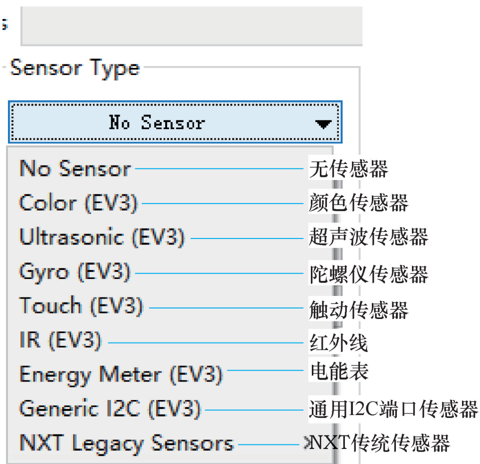


图 1-55

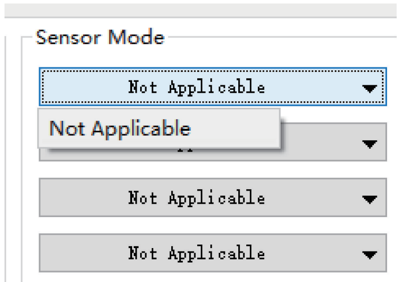


图 1-56

编程完毕后，在将其下载到机器人或虚拟世界之前，必须进行编译程序，这里选择将编译好的程序保存在库－文档之中，如图 1-57 ~ 图 1-58 所示。

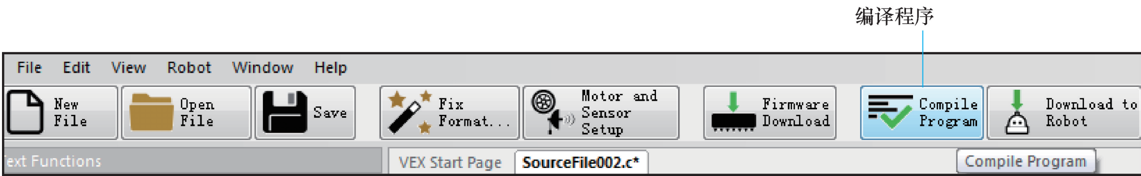


图 1-57

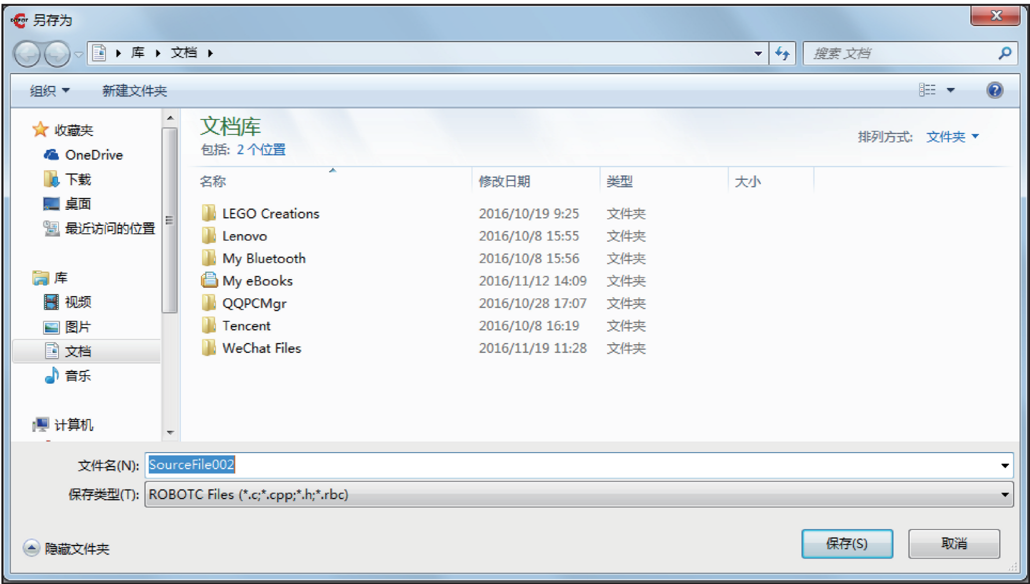


图 1-58

当程序编译完成之后单击 Download to ROBOT（下载到机器人）按钮，根据设定的要求便可以对实体机器人或虚拟世界进行操作了，如图 1-59 所示。

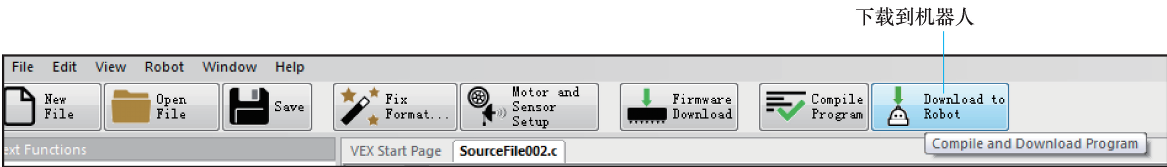


图 1-59

1.3.3 函数库

ROBOTC 中所写的程序里大部分的命令都可以从函数库中找到，无须手动编写，直接拖到编程区即可，例如写一个关于电动机的程序，那么点开 Motors 这一栏就会看到关于电动机的所有语言，此时将需要的程序直接拖拉到编程区进行编写即可，如图 1-60 所示。

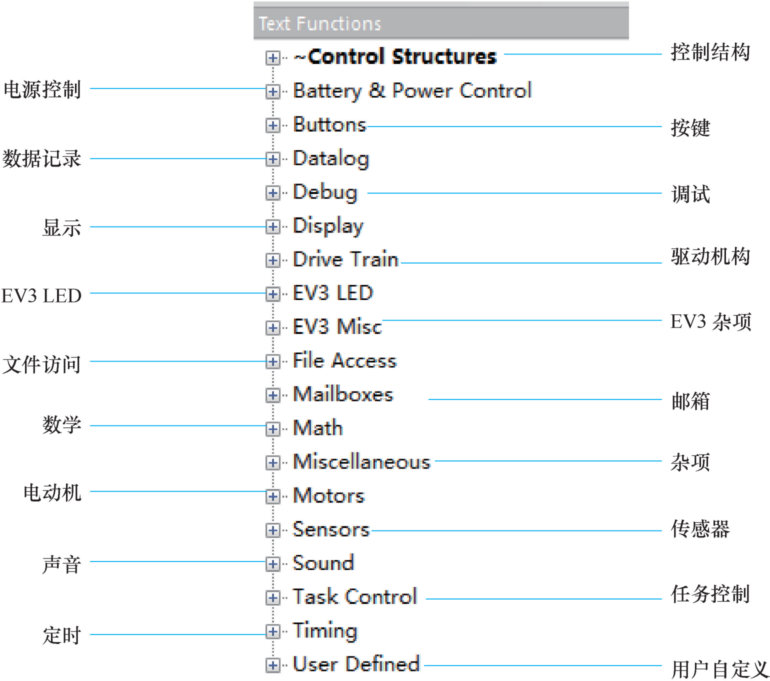


图 1-60

1.4 ROBOTC 虚拟世界

机器人虚拟世界，由 Robomatter 开发，是一个高端的模拟环境，使学生在实体机器人不可用时学习编

程。虚拟世界是 ROBOTC 的一部分，可以将写好的程序在软件中通过虚拟机器人运行。

机器人虚拟世界在 3D 环境中模拟 VEX EDR、VEX IQ 和 EV3 机器人，这些机器人可以使用与实体机器人 ROBOTC 相同的语言进行编程。虚拟世界中有抓手机器人、巡线机器人、自动驾驶机器人等，而场地更是有果园挑战、迷宫挑战、障碍挑战等几十种场地供您选择，它能让用户体验不同机器人的不同挑战。虚拟世界环境完美解决了家庭、教室和其他环境限制条件的问题，例如在教室环境中，机器人虚拟世界可以允许每个学生测试编程概念，而不需要一个随时可用的机器人。

当使用 ROBOTC 时，学生可以在实体和虚拟机器人之间切换，以便在虚拟环境中快速调试和测试他们的代码，迅速体现出所写程序的内容，更方便地找出程序中的不足与误差，然后部署到实体机器人。

机器人虚拟世界允许学生利用虚拟技术在任何地方模拟机器人，在学堂外继续他们的机器人体验。教师可以应用这一点来布置正确的作业，并帮助错过课程的学生。

从虚拟世界中能更好地熟悉 ROBOTC 这个软件，学习不同比赛的规则，让数据更精确，从而更容易掌握 VEX IQ 编程语言。

（1）如果想运行虚拟世界，需要先进行激活，4.55 版本的 ROBOTC 添加许可证后才可以使用虚拟世界，在 ROBOTC 软件中打开 Help（帮助）菜单，选择 Add License（添加许可证）命令，如图 1-61 所示。

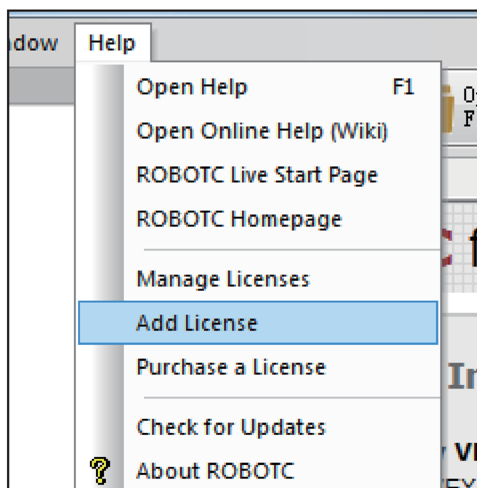


图 1-61

（2）在新窗口中进行选择，有乐高头脑风暴、乐高虚拟世界和 VEX 虚拟世界三个选项，在这里选择乐高虚拟世界，如图 1-62 所示。

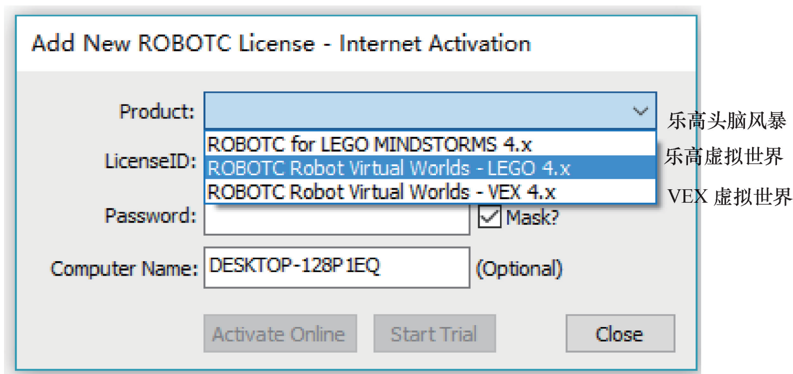


图 1-62

（3）选择乐高虚拟世界之后，在下面两行输入许可证账号与密码，然后单击 Activate Online 按钮进行激活。如果没有购买许可证，那么可以先单击 Start Trial 按钮试用 10 天，如图 1-63 所示。

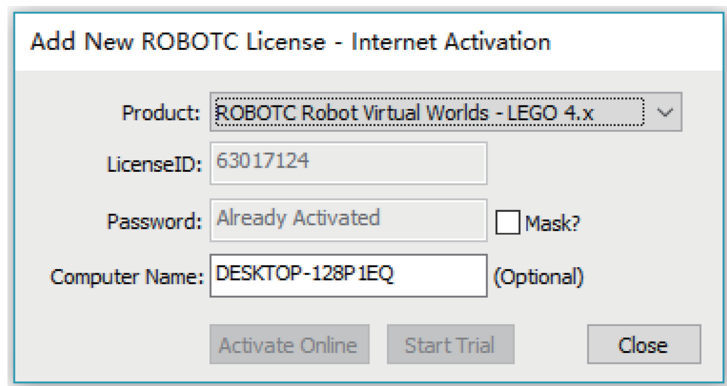


图 1-63

（4）系统会询问是否要创建一个虚拟世界桌面快捷方式，单击“是”按钮即可，如图 1-64 所示。

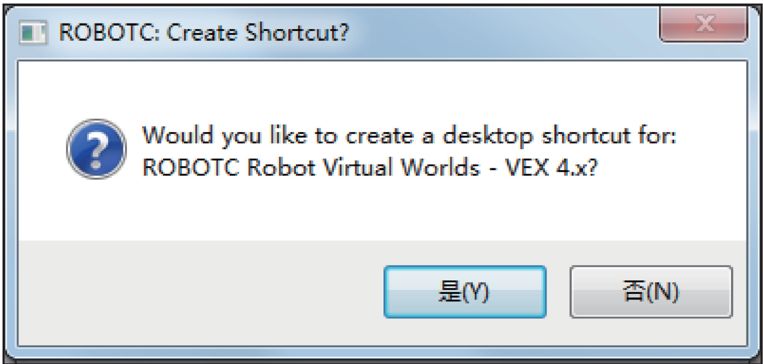


图 1-64

（5）在这里直接单击蓝色选项卡中的 Log In as Guest（游客登录）即可进入虚拟世界主页面，如图 1-65 所示。

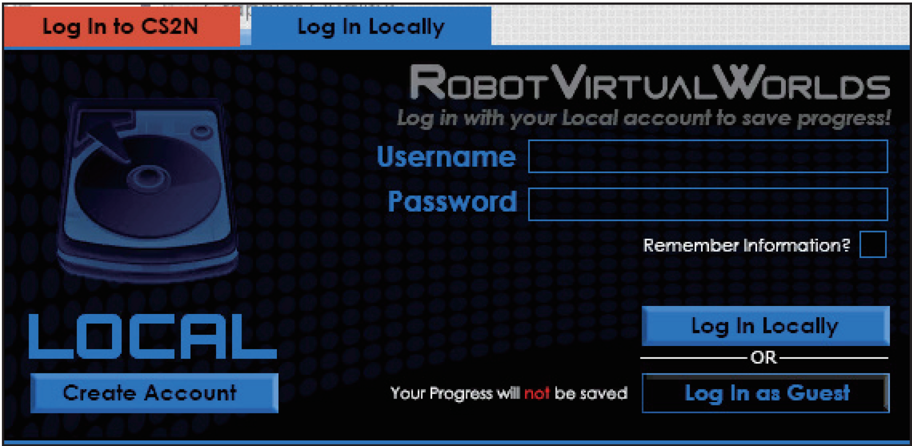


图 1-65

虚拟世界主页面如图 1-66 所示。

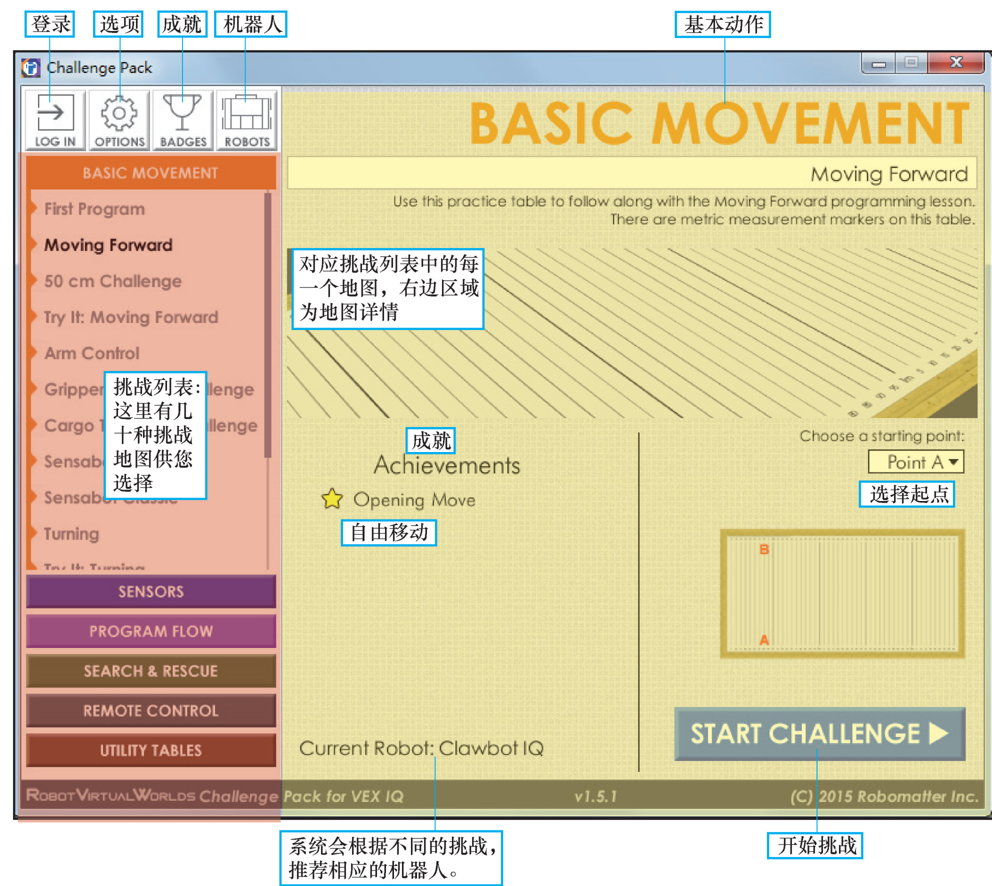


图 1-66

1. 选项

在选项中有如下界面，如图 1-67、图 1-68 所示。

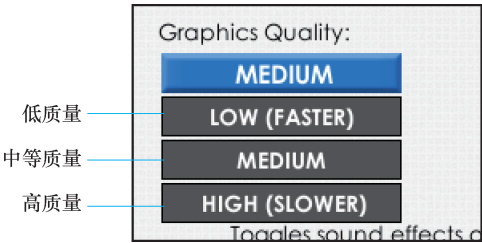

Graphics Quality:
MEDIUM
低质量
LOW (FASTER)
中等质量
MEDIUM
高质量
HIGH (SLOWER)

图 1-67

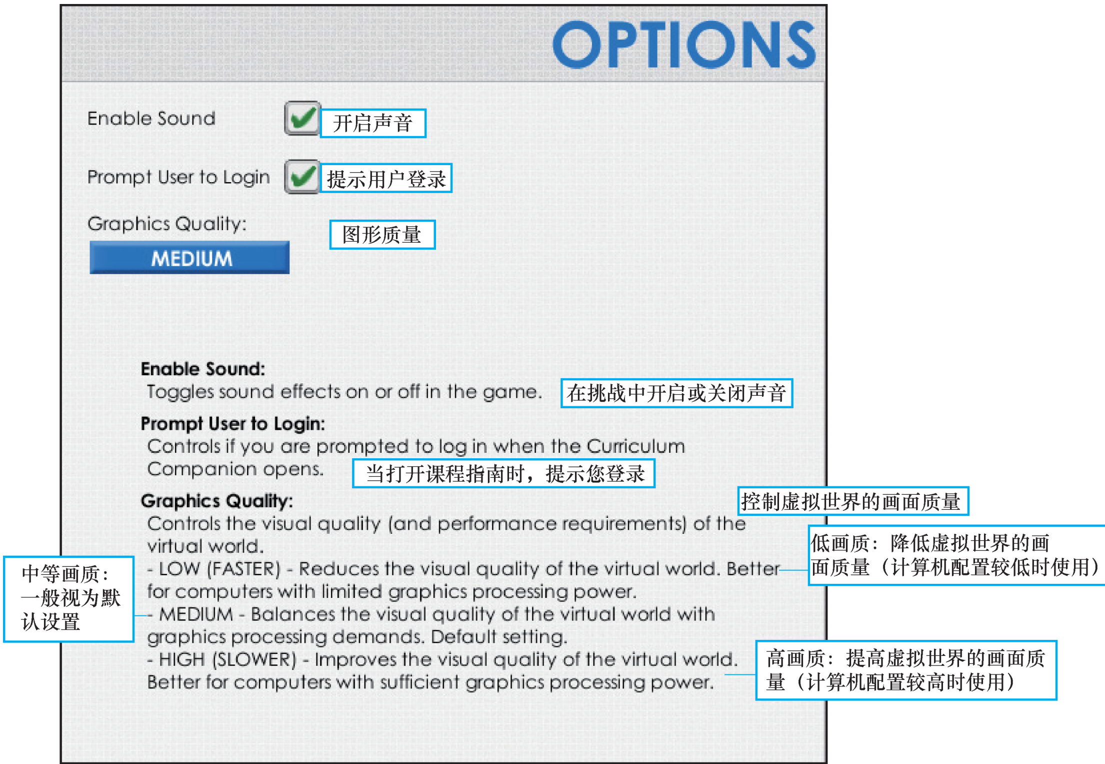

OPTIONS
Enable Sound
开启声音
Prompt User to Login
提示用户登录
Graphics Quality:
图形质量
MEDIUM
Enable Sound:
Toggles sound effects on or off in the game.
在挑战中开启或关闭声音
Prompt User to Login:
Controls if you are prompted to log in when the Curriculum Companion opens.
当打开课程指南时，提示您登录
Graphics Quality:
控制虚拟世界的画面质量
Controls the visual quality (and performance requirements) of the virtual world.
- LOW (FASTER) - Reduces the visual quality of the virtual world. Better for computers with limited graphics processing power.
低画质：降低虚拟世界的画面质量（计算机配置较低时使用）
- MEDIUM - Balances the visual quality of the virtual world with graphics processing demands. Default setting.
中等画质：一般视为默认设置
- HIGH (SLOWER) - Improves the visual quality of the virtual world. Better for computers with sufficient graphics processing power.
高画质：提高虚拟世界的画面质量（计算机配置较高时使用）

图 1-68

2. 成就

每当完成一个挑战，根据完成度的不同，就会解锁相应的成就，在这里可以看到所有成就，如图 1-69 所示。

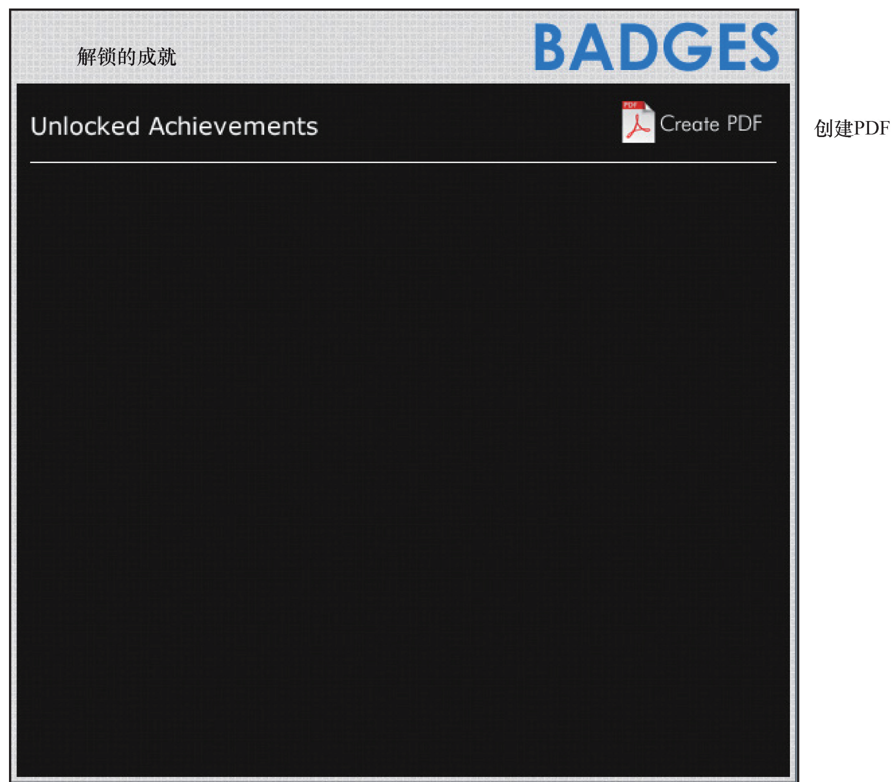


图 1-69

3. 机器人

（1）抓手机器人

在这里列出了常用的几种机器人，每种机器人的传感器不同，设置不同，所发挥的功能也都不同，用它

们分别来挑战不同的地图，如图 1-70 所示。

ROBOTS
EV3
尺寸
Robot Dimensions:
长度 Length: 26 cm
宽度 Width: 15 cm
脚轮 Caster Wheel: 2.2 cm diameter
左轮 Left Wheel: 5.6 cm diameter
右轮 Right Wheel: 5.6 cm diameter

EV3
EV3 - Arm Touch
EV3 - Color Front
EV3 - Dual Color
这里提供了不同的机器人供您选择。

Port Mapping:
手臂电动机 Arm Motor: Port A
左电动机 Left Motor: Port B
右电动机 Right Motor: Port C
触碰传感器 Touch Sensor: Port 1
陀螺仪传感器 Gyro Sensor: Port 2
指示灯传感器 Light Sensor: Port 3
声音传感器 Sonar Sensor: Port 4
手臂电动机编码器 Arm Motor Encoder: Port A
左编码器 Left Encoder: Port B
右编码器 Right Encoder: Port C

开始挑战
START CHALLENGE ▶

图　1-70

选好挑战地图与机器人之后，单击 START CHALLENGE（开始挑战）按钮进入挑战场地，如图 1-71 所示。

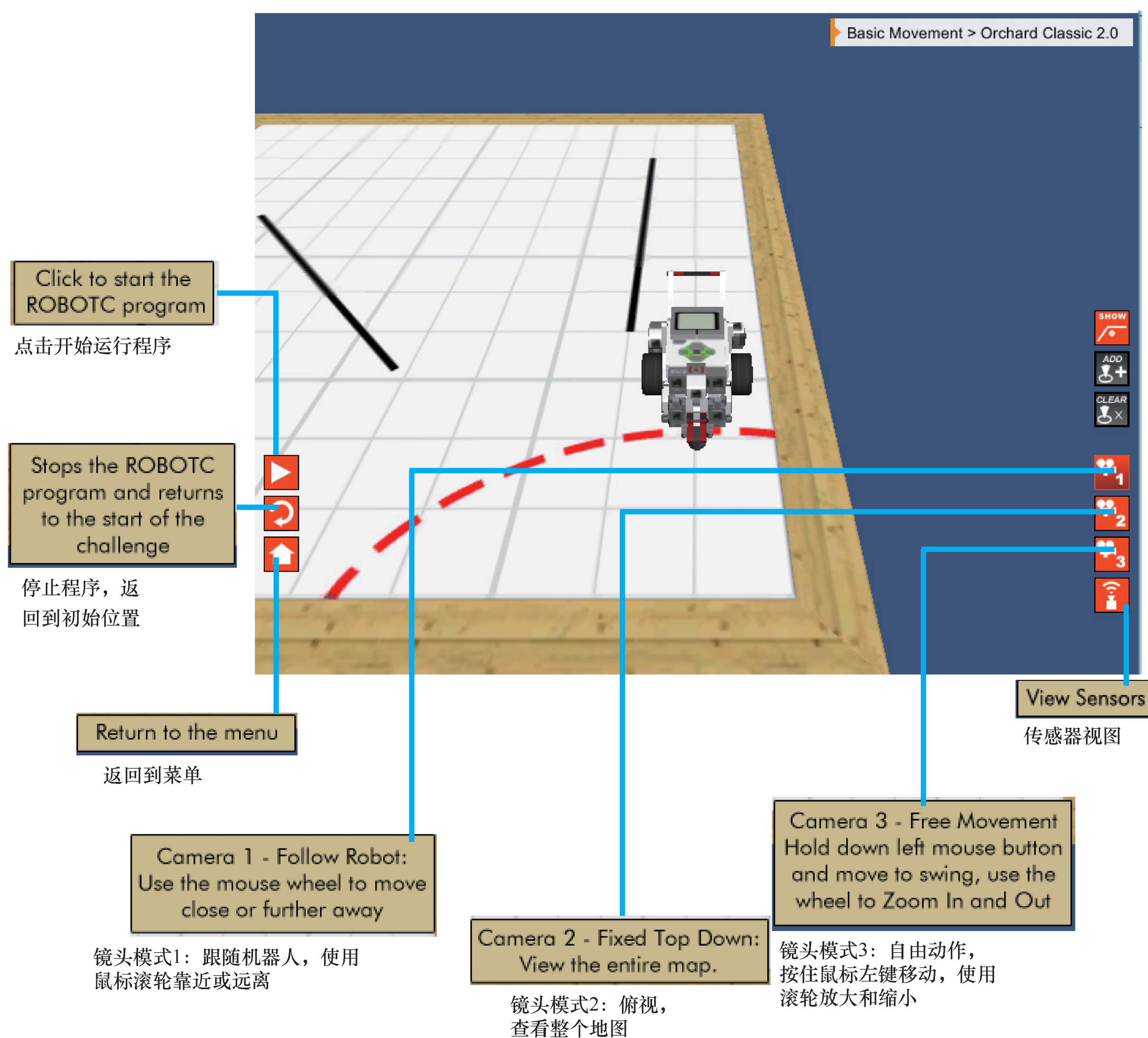

Basic Movement > Orchard Classic 2.0
Click to start the ROBOTC program
点击开始运行程序
Stops the ROBOTC program and returns to the start of the challenge
停止程序，返回到初始位置
Return to the menu
返回到菜单
Camera 1 - Follow Robot: Use the mouse wheel to move close or further away
镜头模式1：跟随机器人，使用鼠标滚轮靠近或远离
Camera 2 - Fixed Top Down: View the entire map.
镜头模式2：俯视，查看整个地图
Camera 3 - Free Movement Hold down left mouse button and move to swing, use the wheel to Zoom In and Out
镜头模式3：自由动作，按住鼠标左键移动，使用滚轮放大和缩小
View Sensors
传感器视图
SHOW
ADD
CLEAR

图 1-71

镜头模式 2 如图 1-72 所示：

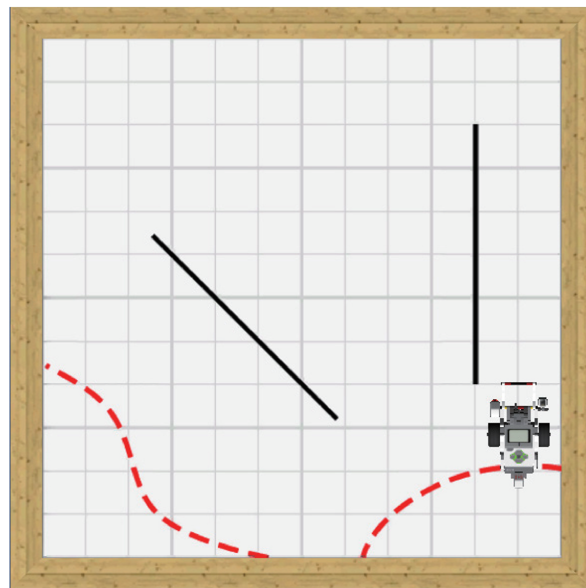


图 1-72

镜头模式 3 如图 1-73 所示：

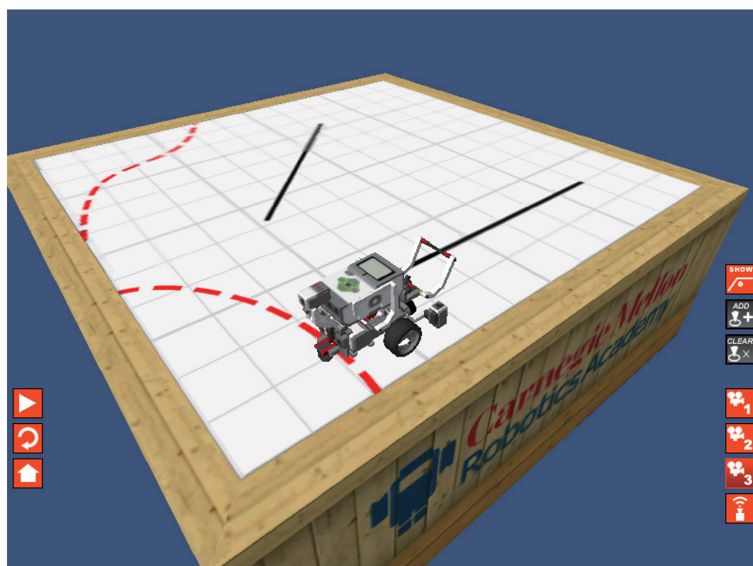


图 1-73

挑战地图中包括基本运动、传感器、程序流程、搜索与营救、多功能赛台五大类，下面分别对这几类进行简单介绍。

（2）基本运动挑战

基本运动顾名思义就是指机器人最基础的一些动作，如前进、后退、转弯等，而挑战地图也是由浅入深，逐渐带着用户熟悉虚拟世界，掌握 ROBOTC 编程，如图 1-74 所示。



图 1-74

（3）传感器挑战

传感器挑战带用户学习各个传感器的功能与用法，通过不同的挑战学习到传感器不同的效果与拓展，如

图 1-75 所示。

SENSORS

一直移动直到碰撞	Forward Until Touch
尝试：向前直到碰撞	Try It: Forward Until Touch
清洁	Vacuum
手臂位置的挑战	Arm Position Challenge
经典手臂位置挑战	Arm Position Classic
一直前进直到靠近	Move Until Near
阈值	Threshold Value
挑战：前进直到靠近	Try It: Forward Until Near
后退直到远离	Backward Until Far
迷宫挑战	Maze Challenge
经典迷宫	Maze Classic
转向角1	Turn For Angle Part 1
转向角2	Turn For Angle Part 2
尝试：转向角	Try It: Turn For Angle
方圈	Square Lap
割草机的挑战	Mower Challenge
经典割草机挑战	Mower Classic
尝试：前进到有颜色	Try It: Forward Until Color
前进到红色	Forward Until Red
前进到停止线	Forward To Stop Line
交通灯挑战	Traffic Lights Challange
经典交通灯挑战	Traffic Lights Classic

图 1-75

（4）程序流程

在这一挑战系列中，将众多程序、动作、传感器等放在一起，设计一个较为复杂的任务，可以更轻松地

学习多种传感器，以及程序之间的配合，从而做出自已所需要的效果，如图 1-76 所示。

	PROGRAM FLOW
环形运动	Looped Movements
方形舞	Square Dance
计数控制环形运动	Loops With Count Control
尝试环形挑战	Try It: Loops
方形舞2	Square Dance 2
传感器控制环形运动	Loops With Sensor Control
方形舞3	Square Dance 3
集装箱装卸挑战	Container Handling Challen
经典集装箱装卸挑战	Container Handling Classic
清理挑战	Move If Clear
颜色传感器对比	Color Sensor Comparison
走迷宫	Maze Runner
尝试if…else语句	Try It: If-Else
走迷宫2	Maze Runner 2
草莓分拣挑战	Strawberry Plant Challenge
经典草莓分拣	Strawberry Plant Classic
障碍物检测	Obstacle Detection
障碍物检测（黑线）	Obstacle Detection Until Bla
穿越果园挑战2.0	Obstacle Orchard Challeng
经典穿越果园挑战2.0	Obstacle Orchard Classic 2.
穿越果园挑战1.0	Obstacle Orchard Challeng
经典穿越果园挑战1.0	Obstacle Orchard Classic 1.
巡线	Line Tracking
尝试：巡线	Try It: Line Tracking
环绕巡线	Line Track For Rotations
巡线挑战	Line Tracking Challenge
经典巡线挑战	Line Tracking Classic

图 1-76

（5）搜索与救援

搜索与救援如图 1-77 所示。



图 1-77

（6）多功能赛台挑战

多功能赛台挑战如图 1-78 所示。



图 1-78

第 2 章　编程讲解与实例

2.1　编程基础讲解

对 ROBOTC 软件有了一个大概的了解之后，就可以开始编程了。

首先，单击工具栏中的 New File 按钮，就会出现编程区，如图 2-1 所示。接下来就在这里编写程序吧。



图　2-1

编程之前先单击工具栏中的 Motor and Sensor Setup 按钮，对电动机（如图 2-2 所示）与传感器（如图 2-3 所示）进行设置，如图 2-2 所示。

（1）为电动机命名时不要取纯数字名字与汉字名字，尽量以英文命名。

（2）电动机型号可以选大型电动机与中型电动机等。

（3）Reversed 如果打勾，电动机会反方向旋转。

（4）PID Control 为默认状态，一般情况下不做任何修改。

（5）Drive Motor Side 是为了方便用户分清电动机安装的左右，所以修改与否影响不大。

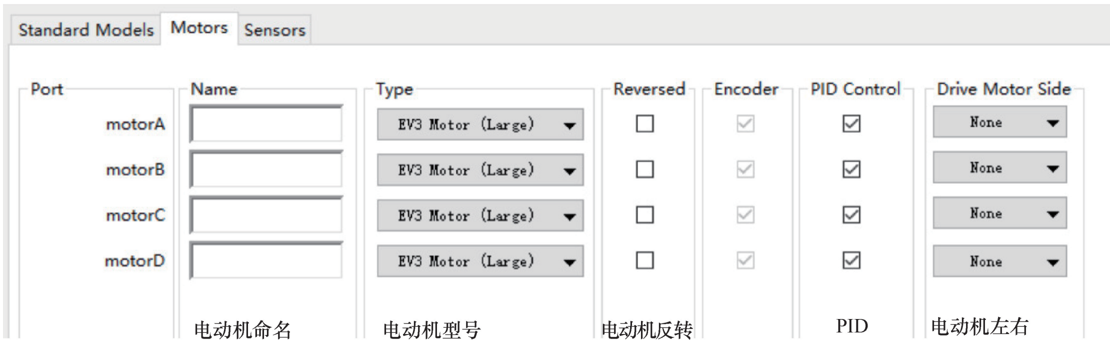


图 2-2

（6）Name：传感器与电动机相同，可以自已随意命名。

（7）Sensor Type：颜色传感器、超声波传感器、触动传感器等。

（8）Sensor Mode：颜色传感器的模式反射光强度、各类颜色等，如图 2-3 所示。

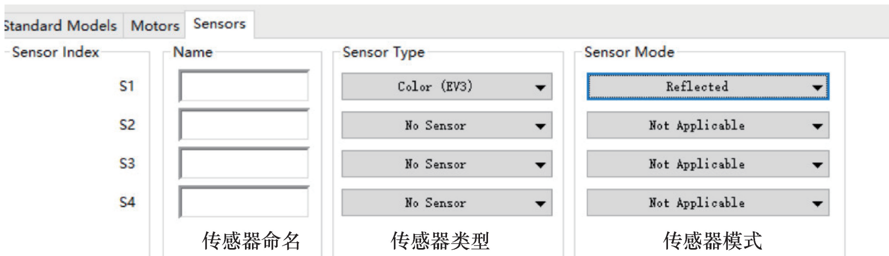


图 2-3

设置好电动机与传感器后，编程区会出现几行预处理命令。这些命令方便用户后续编程，不设置任何传感器也可以，全程只写端口名称，通过代码命令来选择传感器模式一样可以编译。建议每次先设置好电动机

与传感器，以防出错，如图 2-4 所示。

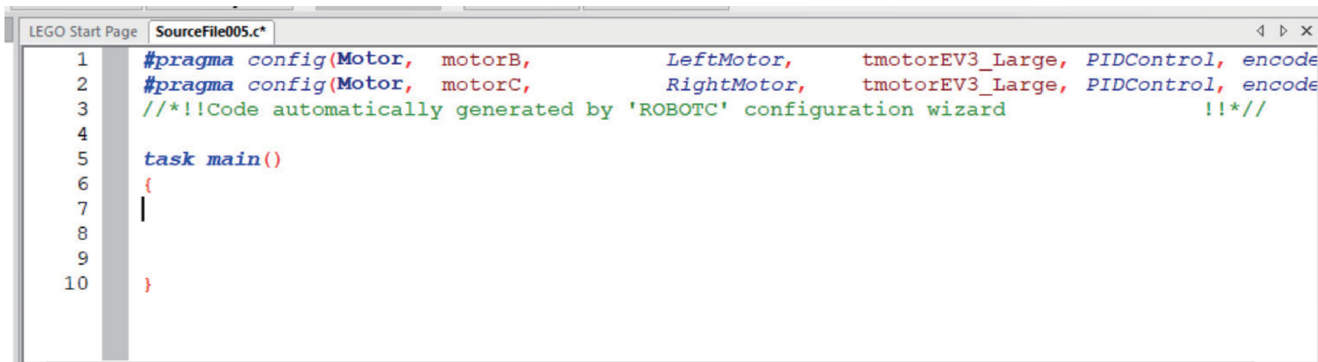


图 2-4

task main 为主函数，所写的程序流程都要写在主函数里面，注意：要写在大括号｛｝之内。变量、常量、自定义的函数等可以写在外面。要设置电动机 B 的功率为 100，要运动，两个电动机需要一起转，所以电动机 C 也是 100，而用 C 语言表示则为：

setMotorSpeed（motorB，100）；

setMotorSpeed（motorC，100）；

以上两行代码设置电动机速度、电动机名称、功率。

注意 C 语言函数名的书写格式，第一个单词全部小写，后面两个单词首字母均为大写，三个单词后紧跟小括号( )，电动机与传感器要执行的命令写在括号中，注意要用逗号“，”隔开，在小括号外再使用分号“；”结尾：setMotorSpeed（motorB，100）；

设置好电动机速度之后，要为其设置运行时间，在这里以 10s 为例，在 C 语言中，时间单位是 ms，所以就是 10000ms，在 C 语言中用 sleep 表示电动机的运行时间，所以应该写为 sleep（10000），如图 2-5 所示。

```
task main()
{
  setMotorSpeed(motorB,100);
  setMotorSpeed(motorC,100);
  sleep(10000);
}
```

图 2-5

注意括号，每一个括号都不是单独的，一定会有一个反括号和它是一对的（不论是小括号、中括号还是大括号，它们总是成对出现的）。

一个函数所带的小括号里面的内容，称为参数，如果有多个参数，需要用逗号分隔（注意所有代码和符号都必须是英文输入，否则会导致软件崩溃，需要重新启动）。

当程序写完之后，单击工具栏中的 Compile Program 按钮，进行编译，系统会将程序保存到指定位置，并且自动检查是否有错，如图 2-6 所示。

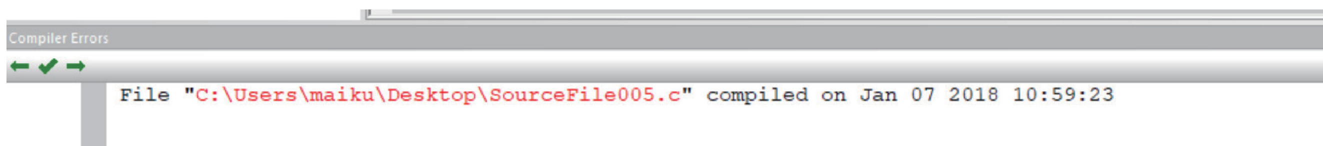


图　2-6

在编译区中，会显示文件保存位置和编译时间，没有提示错误，说明程序语法正确，可以进行下载。下载前需要先选择下载模式，下载到实体机或者虚拟世界（在这里先介绍虚拟世界）。

单击文件栏中的 ROBOT 菜单，选择 Compiler Target/Virtual Worlds 命令，如图 2-7 所示。

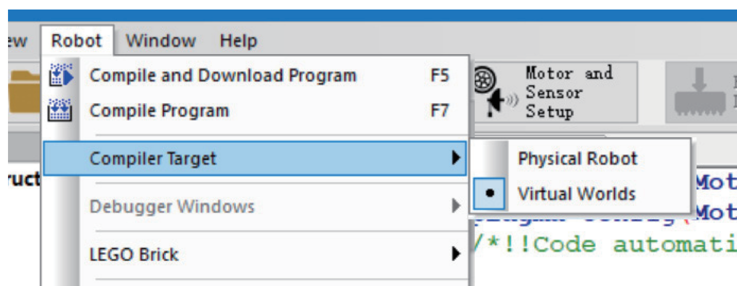


图　2-7

选择虚拟世界的挑战包（可自行下载更多挑战包），单击 Window 菜单，选择 Select Virtual World to Use/Challenge Pack for EV3 命令，如图 2-8 所示。

全部选择完毕之后，单击工具栏中的 Download to ROBOT 按钮，将程序下载到虚拟世界的机器人中，如图 2-9 所示。

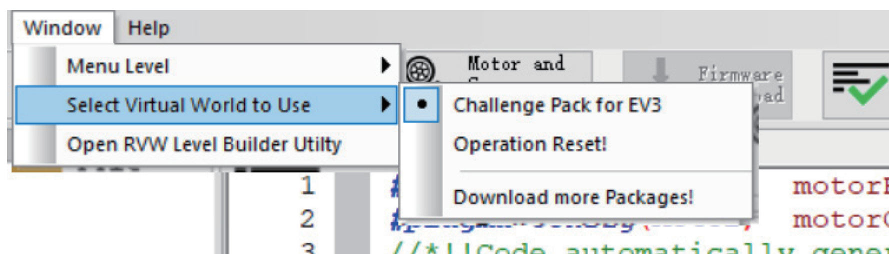


图 2-8



图 2-9

正如前文所述在虚拟世界登录界面单击 Log In as Guest 按钮，即可进入虚拟世界，再选择所用的机器人与地图。

单击工具栏中的 ROBOTS 可以看到机器人信息如图 2-10 所示。

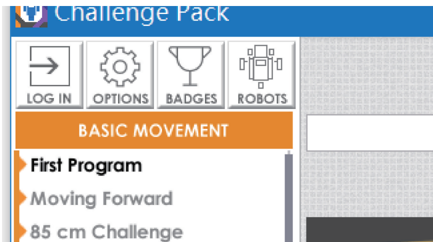


图 2-10

在基础虚拟世界挑战包中有 4 个机器人可以选择，来进行不同地图的挑战任务，如图 2-11 所示。

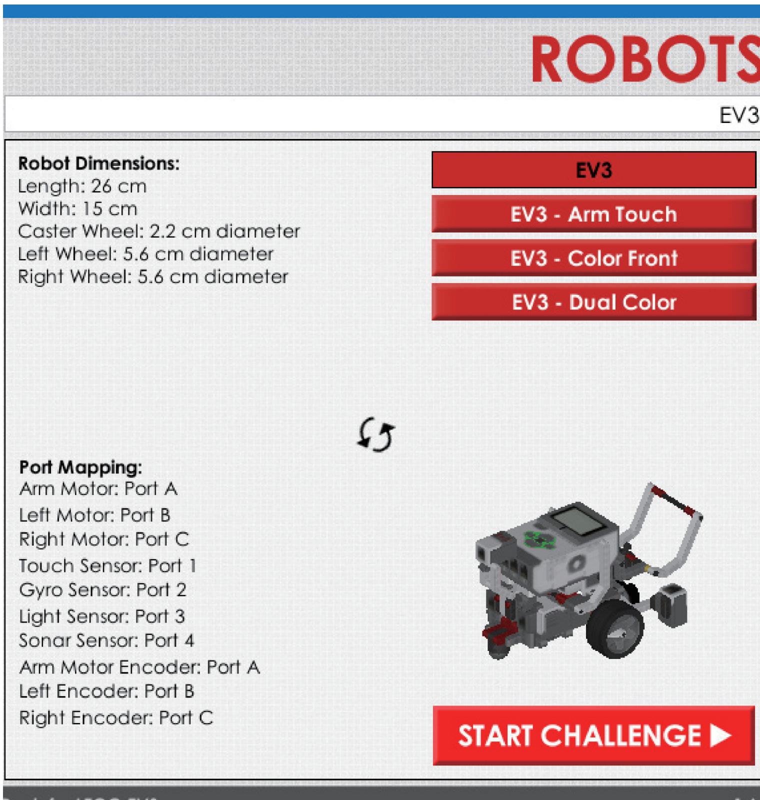


图 2-11

选择好机器人之后，再选择左侧地图，然后单击右下角的 START CHALLENGE 按钮进入挑战任务，如图 2-12。

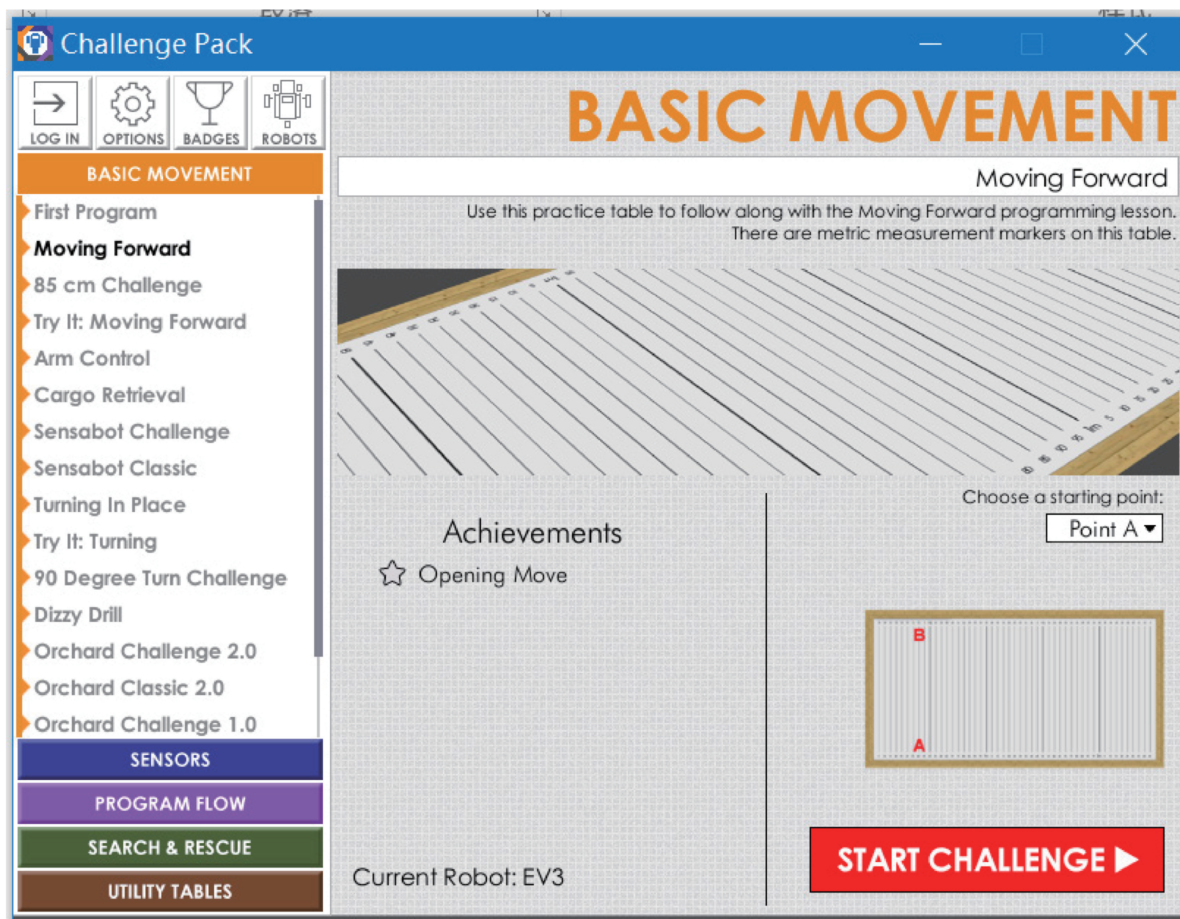


图 2-12

在挑战地图中单击左边的小三角按钮，小车开始运行。如果小车完成所写程序任务，则挑战成功，如果没有完成，则要回去检查在哪一步骤出错了，如图 2-13 所示。

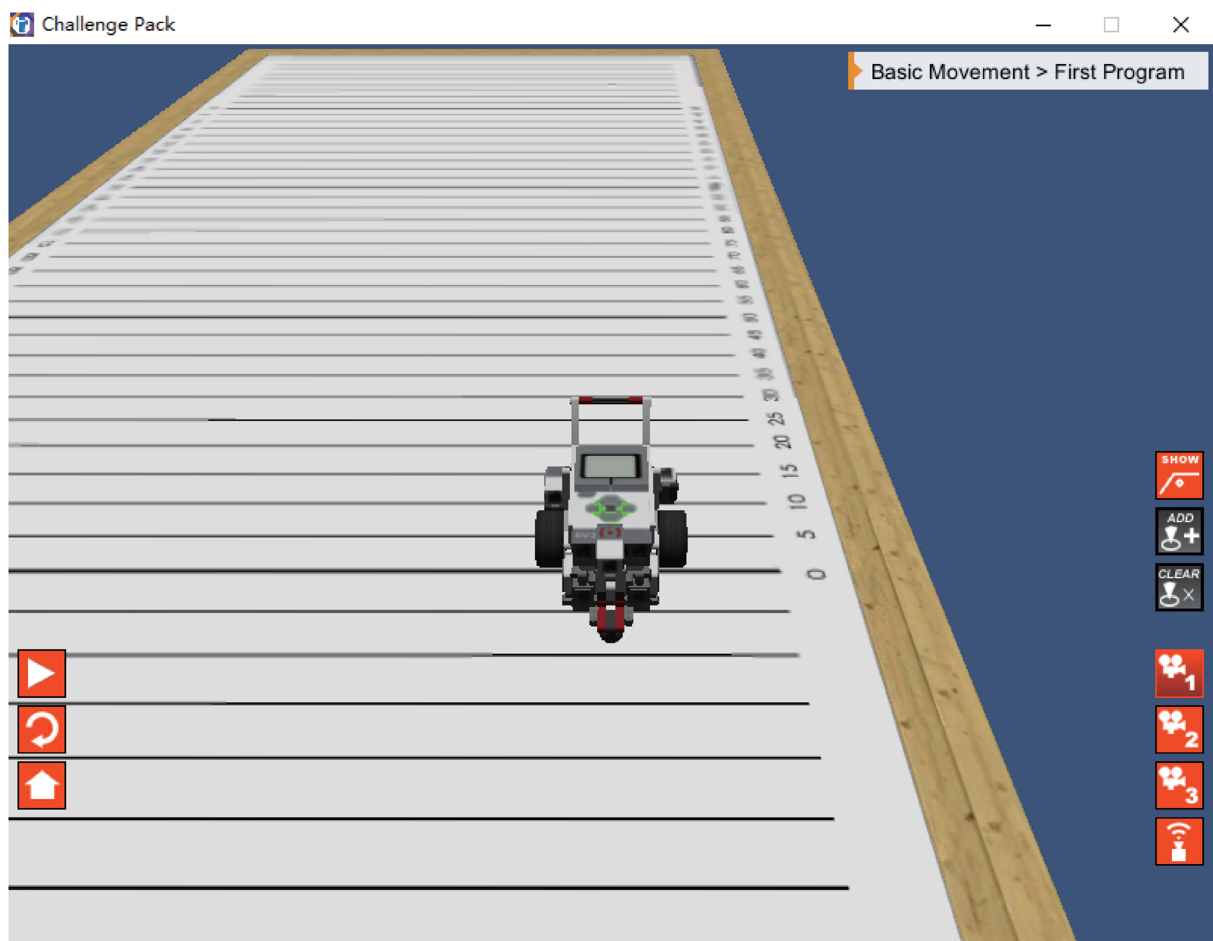


图 2-13

2.2 迷宫挑战

2.2.1 编程知识

```
task main()
```

{

}

task main( ) 意为主要任务，是所有编程开端的固定格式。

{ } 所有的编程语言都必须写到这个大括号中，当出现分级程序时，可在大括号中再加大括号，然后在新的大括号内加入子程序，如下：

{ 主程序

{ 子程序

}

}

当一个程序编写完成时，一定要注意每一组 { } 之间的呼应，以免出现程序不完整而导致程序不能正常运行的情况。

“set”

set 意为设置，一般 set 之后为需要设置的对象，比如电动机、主控器 LED 灯显示等。

例如：

setMotorSpeed(设置电动机速度)

setLEDColor(设置 LED 灯的颜色)

“sleep”

sleep 意为等待，通常跟电动机同时使用，例如电动机以 50 的功率运行 2 s，如下：

```
setMotorSpeed(rightMotor,50);
sleep(2000);
```

“( )” 的用法有以下几种。

(1) 需要对条件进行判断，以执行后续程序，如下：

while(循环条件)

if(判断条件)

(2) 获取信息，如下：

```
getJoystickValue(ChA) > 10
```

“,”一般用在多个参数之间，便于区别不同的参数，例如设置电动机目标时，需要将电动机名称、电动机转动角度以及电动机旋转速度用“,”隔开，如下：

```
setMotorTarget(rightMotor,100,50)
```

“;”每一个完整的小语句之后都以分号结尾（while/if 语句的括号后除外），如下：

```
setMotorSpeed(rightMotor,50);
setMotorSpeed(leftMotor,50);
```

2.2.2 编程指引

想要让小车实现迷宫挑战任务，首先要明白如何让小车实现转向功能，当两个电动机旋转方向相反，或者一个电动机旋转，另一个停止，那么小车就会转向，要如何在编程实现小车转向呢？首先设置所使用的硬件、左右电动机的名称，然后开始编程，如图 2-14 所示。

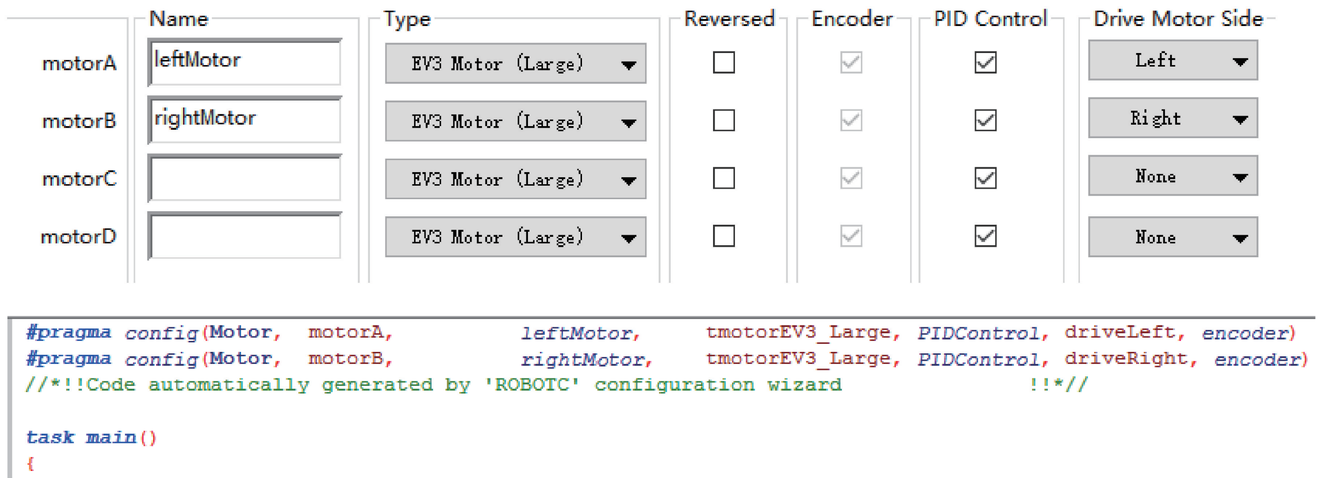


图 2-14

命名完成之后，会自动生成 ROBOTC 配置向导，然后就可以开始编程了。

2.2.3 编程示例

大家可以根据所选地图自行对数据进行修改，第一次编程一定要注意编程格式。

```
task main()                               主函数
{
  setMotorSpeed(leftMotor,100);           设置左电动机速度为100
  setMotorSpeed(rightMotor,100);          设置右电动机速度为100
  sleep(2000);                            这里的单位默认为ms
  setMotorSpeed(leftMotor,100);           设置左电动机速度为100
  setMotorSpeed(rightMotor,-100);         设置右电动机速度为-100
  sleep(2000)
  setMotorSpeed(leftMotor,0);             设置左电动机速度为0
  setMotorSpeed(rightMotor,100);          设置右电动机速度为100
}
```

2.3 电动机控制

2.3.1 编程知识

reset

reset 意为重置，当一个电动机需要根据要求转动一定角度时，通常会使用重置功能，以改变电动机角度为 0 时的初始位置，如下：

```
resetMotorEncoder(rightMotor);
```

注：一般重置电动机时要写 MotorEncoder（电动机编码器）

陀螺仪根据程序要求也常用到重置功能，如下：

```
resetGyro(gyroSensor);
```

```
Target
```

Target 在这里为设置电动机的目标，使电动机运行的距离更精确，一般与 set 组成完整语句，例如设置电动机以 50 的功率旋转 600°，如下：

```
setMotorTarget(rightMotor,600,50);
```

```
waitUntil
```

waitUntil 意为一直等到……的时候，通常将 waitUntil 用在一段程序的结尾，表示将此段程序运行完毕之后，再运行下一个程序，例如一直到右电动机停止旋转之后，左电动机开始旋转，如下：

```
setMotorTarget(rightMotor,600,50);
waitUntilMotorStop(rightMotor);
setMotorTarget(leftMotor,600,50);
waitUntilMotorStop(leftMotor);
```

2.3.2 编程指引

有没有什么好的方法可以使小车精确行驶 50cm 就停止呢？本次挑战学习一下编码器的用法。如果想准确到达指定位置，就需要知道电动机当前的位置，该如何获取呢？乐高电动机中有专门记录角度的传感器：编码器。

编码器原理：

（1）如何发现旋转（光电传感器与码盘）

（2）如何计数（每一个码盘上的间隙）

（3）如何确定正反转（方波相位差）

可以用小车电动机旋转一周，测量小车前进距离，再用 50cm 除以所测量的距离，得出小车行驶 50cm 所旋转的角度与旋转一周角度的倍数，再乘以 360°即为小车行驶 50cm 电动机所旋转的角度。

2.3.3 编程示例

编程示例如下。

```
task main()
{
resetMotorEncoder(leftMotor);
resetMotorEncoder(rightMotor);
setMotorTarget(leftMotor,700,100);
setMotorTarget(rightMotor,700,100);
waitUntilMotorStop(leftMotor);
waitUntilMotorStop(rightMotor);
}
```

重置电动机编码器。

给电动机设置终点的控制语句需填入所需旋转角度（700°只是示例，自已进行调整）。

WaiUntil的语句可以保证在电动机执行完编程语句后才会停止程序。

2.4 触动传感器

2.4.1 编程知识

```
while
{
}
```

while 不仅仅在 ROBOTC 编程中常用，在 C 语言编程整个系统中都是常用语句，意为“当……的时候”。一般情况下，while 后面都会跟有大括号，如果想使一些语句在满足某一条件时循环执行，那么可以使用 while 语句。正常的格式如图 2-15 所示。

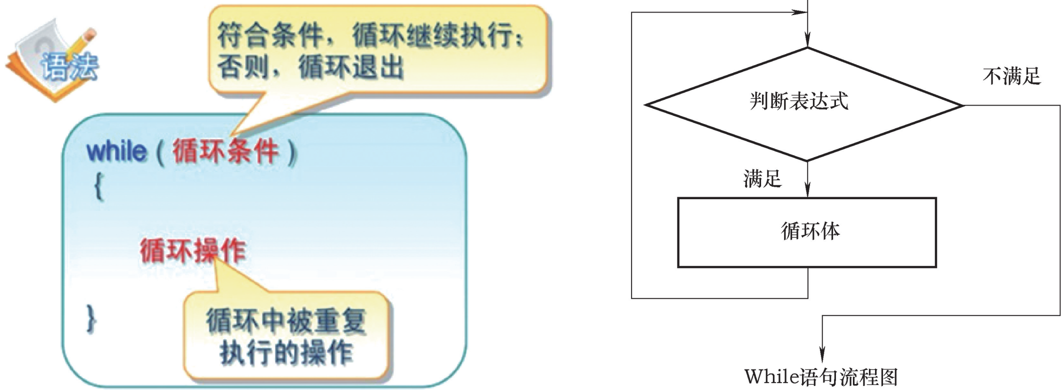


图 2-15

while （true）是一个无限循环，因为表达式的值一直为真。

while （1）和 while （true）一样，均为无限循环。

“= =”表示的是数学中的等号，客观地体现传感状态信息；而“=”表示的是赋值，主观地赋予传感器一个数值，例如，a = =1 表示 a 等于 1。a =1 表示把 1 赋值给变量 a。

```
(SensorValue(bumper) = =0)
//触碰传感器的值为 0(触碰传感器未被按压)
(getColorName(c1) = =colorRed)
//颜色传感器检测到颜色为红色
```

“get”

get 意为获取，常用于判断条件语句，一般放在 while 或 if 后的括号内使用，例如当检测到的颜色为……，如下：

```
if(getColorName)
```

注意：在这里的拼写方式，三个单词之间没有空格，而第一个单词的首字母为小写，后面两个单词的首字母均为大写，不仅仅用于 get 语句，例如 set、reset、wait 等，都用此固定格式书写。

“Value”

Value 意为数值，常用于判断条件语句中，与 get 组成完整语句，获取传感器当时的数值，例如获取触碰传感器的数值为 0（触碰传感器被按压），如下：

```
(getSensorValue(bumper) = =0)
```

“repeat(n)”

repeat 意为重复，根据程序要求，可以命令 repeat 后的程序重复 n 次，比如让电动机以 50 的功率旋转 2s，重复 4 次，如下：

```
task main()
{
 repeat(4)
   {
     setMotorSpeed(rightMotor,50);
```

```
        sleep(2000);
      }
  }
```

2.4.2 编程指引

什么是传感器？触动传感器相当于触觉，以外界环境对自已的碰撞情况进行判断，是机器人接受外界信息的窗口——触动传感器感受到的是按压信号。该如何使触动传感器语句对电动机进行控制呢？

getTouchValue(name);

语句获取的信息为按或没按（即是“真”与“伪”）

```
while( 循环条件 )
{
内容;
}
```

当循环条件为真时，则循环内容，每循环一次就重新判断一次条件，当条件为伪时，则跳过循环。试着编程实现在虚拟世界中完成任务：until touch（碰触 4 面墙）。

2.4.3 编程示例

编程示例如下。

```
task main()
{repeat(4)     重复四次
  {
while(getTouchValue(S1)==0) 触动传感器未被按压
{
setMotorSpeed(motorB,50);   小车向前行走
setMotorSpeed(motorC,50);

}
setMotorSpeed(motorB,-50);  当触动传感器被按压，
setMotorSpeed(motorC,0);    则跳出循环执行当前命
sleep(700);                 令，小车转弯

}
}
```

触动传感器还可以实现什么功能呢？可以编程把触动传感器作为开关，每次按下可以在移动和停止之间进行切换。

2.5 超声波传感器

2.5.1 编程知识

“<”，“>”表示小于或大于某个值，常用于判断条件语句中，通常使用在摇杆、陀螺仪与超声波传感器的程序中，如下（摇杆值大于 10）：

```
getJoystickvalue(chA)>10;
```

视觉可以帮人类躲避障碍，那么机器人该凭借什么躲避障碍呢？由于超声波指向性强，能量消耗缓慢，在介质中传播的距离较远，如图 2-16 所示，因而经常用于距离的测量，如测距仪和物位测量仪等都可以通过超声波来实现。利用超声波检测往往比较迅速、方便、计算简单、易于做到实时控制，并且在测量精度方面能达到工业等级的要求，因此在移动机器人研制上也得到了广泛应用。

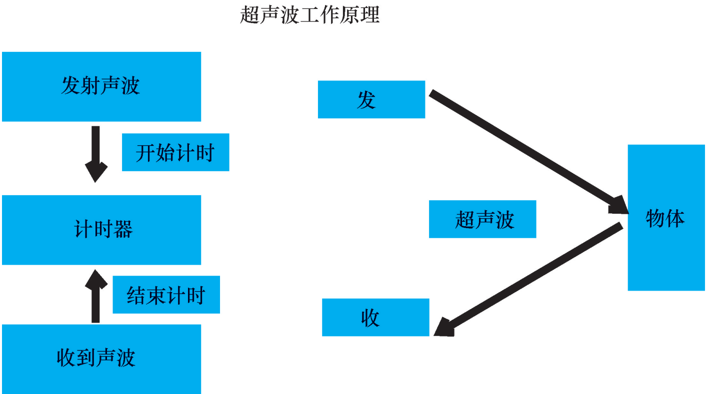


图 2-16

2.5.2　编程指引

getUSDistance(name);

超声波传感器所获得的数据，来自对时间间隔的计算，所以和触动传感器不同的是，结果不是真伪两种，而是连续变化的数值，距离越大数值越大。

试着编程实现在虚拟世界中完成任务：move until near，通过编程来实现在距离目标的 40～100cm 处来回行进。

2.5.3　编程示例

编程示例如下。

```
task main()
{
  while(getUSDistance(X)>100)
{
  setMotorSpeed(motorB,40);
  setMotorSpeed(motorC,40);
}
}
```

当超声波传感器探测到100cm之内有障碍物时跳出循环。

2.6　陀螺仪传感器

2.6.1　编程知识

“degrees”

degrees 意为角度，电动机的角度用 target 来表示，而陀螺仪的旋转角度则用 degrees 来表示，常用于判断

条件语句，一般与 get 组成完整语句，比如当陀螺仪旋转大于 90°时，如下：

```
while(getGyroDegrees(gyroSensor)>90)
```

2.6.2 编程指引

动物可以根据自身感觉把握平衡，那么机器人通过什么得知自身是否平衡呢？本次挑战带大家一起学习陀螺仪传感器的使用。

陀螺仪的原理就是，一个旋转物体的旋转轴所指的方向在不受外力影响时，是不会改变的。人们根据这个道理，用它来保持方向，然后用多种方法读取轴所指示的方向，并自动将数据信号传给控制系统。骑自行车其实也是利用了这个原理。轮子转得越快越不容易倒。

```
getGyroDegrees(name);
```

陀螺仪传感器获取的数值。

试着编程来实现在虚拟世界中完成任务：90 degrees turn challenge。

2.6.3 编程示例

编程示例如下。

```
task main()
{
  while(getGyroDegrees(S)>-60)     //当陀螺仪传感器旋转角度小于-60°
{                                  //时电动机进行转弯。
  setMotorSpeed(A,30);
  setMotorSpeed(B,-30);
}
}
```

2.7　颜色传感器

2.7.1　编程知识

```
if
...
else{
}
```

if 语句跟 while 类似，也是 C 语言中的常用语句，意为“如果”。

if...else 语句用来判定所给定的条件是否满足，根据判定的结果（真或假）决定执行哪个语句。

if 语句使用分为三种情况：

（1）单独使用 if 语句

（2）使用 if...else 语句

（3）使用多个 if...else 语句

1. 单独使用 if 语句

有些时候，需要在满足某种条件时执行一些操作，而不满足条件时，就不进行任何操作，这个时候可以只使用 if 语句。也就是说，if 和 else 不必同时出现。

单独使用 if 语句的形式为：

```
if(判断条件)
{
    语句块
}
```

如果判断条件成立就执行语句块，否则直接跳过。其执行过程可表示为图 2-17。

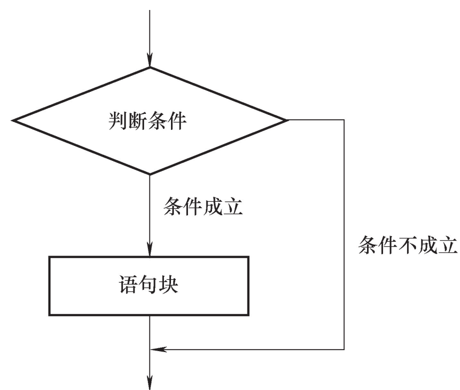


图 2-17

2. 使用 if...else 语句

if 意为“如果”，else 意为“否则”，用来对条件进行判断，并根据判断结果执行不同的语句。总结起来，if...else 的结构如图 2-18 所示。

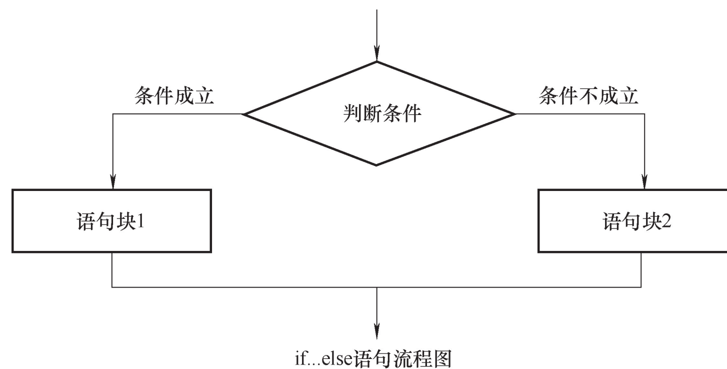


图 2-18

```
if(判断条件)
{
    语句块 1
}else
```

```
{
    语句块 2
}
```

如果判断条件成立，那么执行语句块 1，否则执行语句块 2。其执行过程可表示为上图。

3. 使用多个 if...else 语句

if...else 语句也可以多个同时使用，构成多个分支，形式如下：

```
if(判断条件 1)
{
    语句块 1
} else if(判断条件 2)
{
    语句块 2
}else if(判断条件 3)
{
    语句块 3
}else if(判断条件 m)
{
    语句块 m
}else
{
    语句块 n
}
```

从上到下依次检测判断条件，当某个判断条件成立时，执行其对应的语句块，然后跳到整个 if...else 语句之外继续执行其他代码。如果所有判断条件都不成立，则执行语句块 n，然后继续执行后续代码。也就是说，一旦遇到能够成立的判断条件，则不再执行其他的语句块，所以最终只能有一个语句块被执行。其执行

过程可表示如图 2-19 所示。

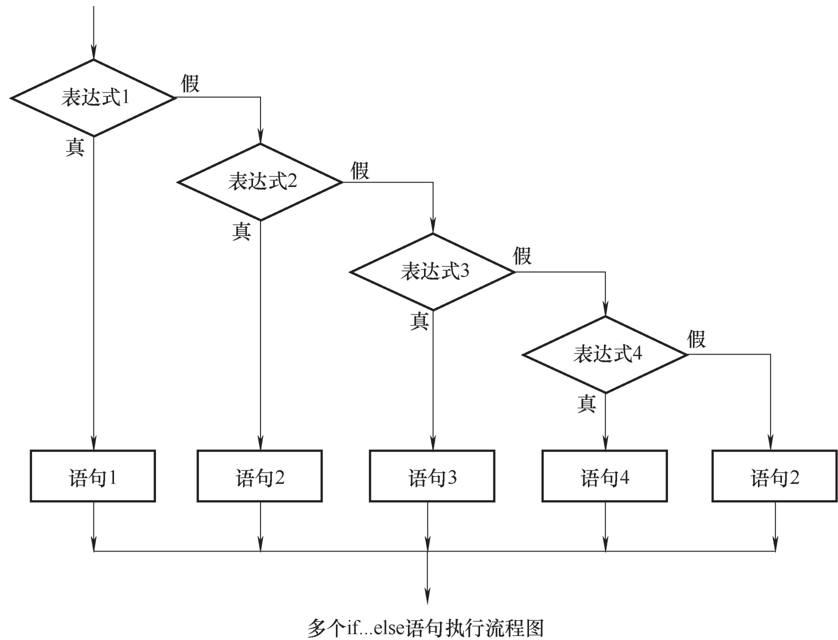


图　2-19

2.7.2　编程指引

颜色传感器是一种传感装置，是将物体颜色同已经输入过的参考颜色进行比较来检测颜色的装置。当两个颜色在一定的误差范围内吻合时，输出检测结果。任何颜色的可见光都可以分解成 RGB（红绿蓝）三种颜色，那么根据三种颜色反射光值的比例就可以判断出当前检测到的颜色了。

```
getColorName(colorRed);
```

获取颜色数据：

颜色值格式：colorRed、colorBlue...

试着编程来实现在虚拟世界中完成任务：forward until red。

2.7.3 编程示例

编程示例如下。

```
task main()
{
while(1)
{
setMotorSpeed(B, 40);
setMotorSpeed(C,40);
while(getColorName(E)==colorBlue)
{
setMotorSpeed(B, 0);      当颜色传感器检测到蓝色
setMotorSpeed(C,0);       时，小车停止前进
}
}
}
```

如果想要程序在两种以上的情况中切换，可以使用 if... else，如下。

```
task main()
{
while(1)
{
if(getColorName(E)==colorBlack)
{setMotorSpeed(B,0);
  setMotorSpeed(C,0);
}
else
{
setMotorSpeed(B,100);
setMotorSpeed(C,100);
}
}
}
```

动手搭建小车并编程来实现将成熟和未成熟的草莓分类（红色和绿色乐高块，分为左、右两边）。

这里为方便编程使用一个函数，这里先做简单了解。使用 void 函数可以简单控制多个电动机。

```
void allmotor(int a,int b,int c)
{       建立函数，三个电机
  setMotorSpeed(arm,a);
  setMotorSpeed(left,b);
  setMotorSpeed(right,c);
}
task main()
{while(1)
  {
    while(getUSDistance(USsensor)>100)
    {
      allmotor(0,30,30);  控制函数，直接对电动机进行操作。
    }
    allmotor(20,0,0);
    sleep(500);
    if(getColorName(Colorsensor)==colorGreen)
    {
      allmotor(0,30,-30);
      sleep(1000);
      allmotor(-20,0,0);
      sleep(500);
      while(getGyroDegrees(gyro)<=0)
        allmotor(0,-30,30);
    }
    else
    {
      allmotor(0,-30,30);
      sleep(1000);
      allmotor(-20,0,0);
      sleep(500);
      while(getGyroDegrees(gyro)>=0)
        allmotor(0,30,-30);
    }
  }
}
```

2.8 巡线

2.8.1 编程指引

如何让小车沿着弯曲的线前进呢？其实巡线不需要确定颜色，只需要知道明暗即可。那么该如何获得检测目标颜色的明暗呢？根据反射光的强度，上一次介绍的颜色传感器是通过 RGB 三种值来计算颜色的，那么 LEGO 的颜色传感器是如何检测反射光强度的呢？仔细观察检测反射光强度时传感器发出的光，就可以得到答案了。只发出一种光并检测反射回来多少，就可以确定目标颜色深浅了。

```
getColorReflected(nDeviceIndex);
```

以上代码获取反射光强度，语句获得的数据类型为由暗到亮连续变化的数字。

2.8.2 编程示例

试着编程来实现在虚拟世界中完成任务：Line Tracking，车辆在虚拟世界 Line Tracking 地图上停止与运动，同时打开 debugger 窗口中的 sensor，监测颜色传感器在黑线上与白色地方的数值。

```
task main()
{
while(1)
{
if(getColorReflected(G)<50)
{setMotorSpeed(B,10);
setMotorSpeed(C,0);
}
else
{setMotorSpeed(B,0);
  setMotorSpeed(C,10);
  }
}
}
```

当颜色传感器检测到灰度值小于50时，小车左转，否则（大于50）小车右转，如此循环，小车便可以沿着黑线前行。

```
while(getMotorEncoder(MB)<360)
{}
```

以上代码使用编码器在走出一定距离后切换左右巡线。

```
task main()
{
while(getMotorTarget(C)==720)
{
if(getColorReflected(G)<50)
{setMotorSpeed(B,10);
setMotorSpeed(C,0);
}
else
{setMotorSpeed(B,0);
  setMotorSpeed(C,10);
  }
}
while(1)
{
if(getColorReflected(G)<50)
{setMotorSpeed(B,10);
setMotorSpeed(C,0);
}
else
{setMotorSpeed(B,0);
  setMotorSpeed(C,10);
  }
}
}
```

2.9 显示与 LED

1. 显示一段文本

```
displayTextLine( LineNumber,"语句");
```

lineNumber 表示是要将语句显示在第几行；这里要显示的语句属于字符串，所以要用双引号括起来。

例如：在屏幕第一行显示：What's your name?

```
task main ()
{
displayTextLine(1,"What's your name?");
}
```

2. 显示参数数值

```
displayTextLine(Linenumber,"%d",value)
```

%d 是格式说明符号，在输入输出时，对不同类型的数据（如 int，float，char 等）要使用不同的格式说明：

%d，用来输出十进制整数。

%f，用来输出实数（包括单、双精度），以小数形式输出。

%c，用来输出一个字符。

%s，用来输出一个字符串。

Value 是参数数值。

例如机器人向后倒退，同时在屏幕的第三行将超声波检测到的距离显示出来。

```
task main()
{
while(1)
{
```

```
setMotorSpeed(motor1,-50);
setMotorSpeed(motor6,-50);
displayTextLine( 3,"%d", getDistanceValue(DistanceSensor ));
}
}
```

3. 显示图形

（1）显示直线

drawLine（$x1$，$y1$，$x2$，$y2$）;

（$x1$，$y1$）为起点坐标，（$x2$，$y2$）为终点坐标，通过这两点连成一条线段。

（2）显示圆形

drawCircle（x，y，r）;

（x，y）是圆心坐标，r 为圆的半径，单位为像素。

试着让屏幕显示一些其他图形，例如画一副眼镜，画一个糖葫芦……

例如：

drawLine（20，20，30，30）;

drawCircle（20，20，10）;

LED 灯指的是 EV3 主控器上可以闪烁的指示灯，通过编程可以让指示灯闪现不同的颜色。

LED 灯语句：

setLEDColor（ledColor）

可选方式分为常亮、闪烁和脉搏式闪烁。颜色可选红、橘、绿三种颜色。

ledRed（长亮）

ledRedFlash（闪烁）

ledRedPulse（脉搏式闪烁）

试着使用 LED 灯模拟交通灯，要求时间合理，加入闪烁。

```
task main()
{
  while(1)
  {
setLEDColor(ledGreen);
sleep(10000);
setLEDColor(ledGreenPulse);
sleep(2000);
setLEDColor(ledOrangeFlash);
sleep(2000);
setLEDColor(ledRed);
sleep(10000);
setLEDColor(ledRedPulse);
sleep(2000);
}
}
```

设置主控器 LED 灯为红色，亮10s，然后红色闪 2s。

2.10 变量

2.10.1 编程知识

int

int 意为赋值。这个数值有可能会随着外界因素而改变（例如触碰传感器的按压次数），用 int 来代表这个数值，编写到程序中，例如为触碰传感器赋予一个变量数值，来记录触动传感器被按压的次数，如下：

```
int a =0;
while(1)
{
while(getTouchLEDValue(TouchLED1) = =0)
{
```

```
sleep(50);
} a++;
while(getTouchLEDValue(TouchLED1)==1)
{
sleep(500);
}
}
```

以上代码设置变量 a，初始数值为 0，触动传感器每被按压一次，a 的数值就会加 1。

变量来源于数学，是计算机语言中能存储计算结果或能表示值的抽象概念。变量可以通过变量名访问。学习变量，首先要了解 C 语言的数据类型，如图 1-20：

C 语言包含的数据类型如下图所示：

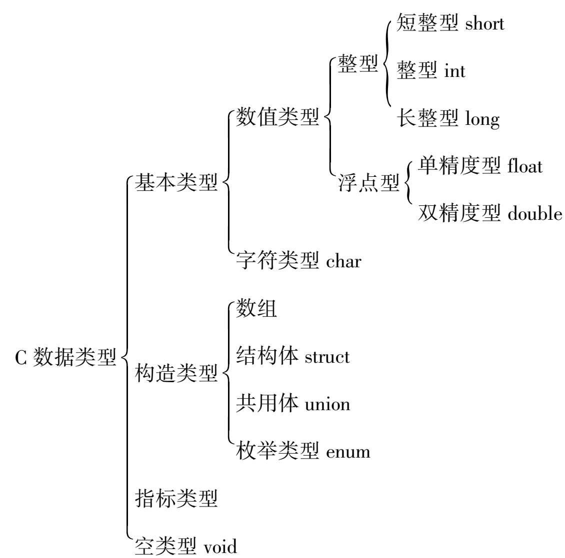


图 2-20

同时还要明白 C 语言中运算符的用法如图 2-21 所示。

名称	运算符号	举例
加法运算符	+	2+10=12
减法运算符	-	10-3=7
乘法运算符	*	2*10=20
除法运算符	/	30/10=3
求余运算符（模运算符）	%	23%7=2
自增运算符	++	int a = 1; a++
自减运算符	--	int a = 1; a--

图　2-21

2.10.2　编程指引

变量的定义方法是：先定义，后赋值。

例如：

```
int i ;
i =0;
```

也可以在定义时就赋值。

例如：

```
    int i =0;
```

先看一下如何让机器人逐渐加速前进。

```
task main()
{
    int i =0;     先定义变量 i =0
    while( i <100)
    {
    setMotorSpeed( leftmotor,i );
    setMotorSpeed(rightmotor,i );    设置电动机速度为变量 i
    sleep(10);
```

```
    i + + ;
    }
}
```

每隔 10ms 变量 i 便会加 1，如此循环便实现小车均匀加速前进。

2.10.3 编程示例

（1）尝试显示变量内的数字。

displayTextLine（ 行数 ，“输出进制”，内容 ）；

若要输出十进制整数格式，应使用 “%d”。

```
task main()
{
  int i;
  i=1;
  displayTextLine(1,"%d",i);
}
```

（2）显示从 0 到 100 之间所有的自然数。

```
task main()
{
int i=1;
while(i<100)
{
displayTextLine(2,"%d",i);
sleep(500);
i=i+1;
}
}
```

试着使小车在 5s 内从静止加速到最快速度，并在 5s 后减速到静止。

```
task main()
{
int i=0;
while(i<100)
 {setMotorSpeed(motorA,i);
  setMotorSpeed(motorB,i);
  sleep(50);
  i++;
 }
  while(i>0)
 {setMotorSpeed(motorA,i);
  setMotorSpeed(motorB,i);
  sleep(50);
  i--;
 }
}
```

变量i从 0到100 需要循环 100 次，所以这里时间取5s的百分之一，也就是50ms。

2.11 声音

EV3 可以发出不同的音调，甚至播放歌曲，如何编程实现呢？让我们一起学习一下吧。

播放音调的语言：

playTone(Hz, time);/ * Hz 是频率,time 是时间,100 表示 1s * /

播放音频的语言：

playSoundFile(“soundFileName");

例如：

playSoundFile(“EV3.rsf”);

设置音量的语言：

setSoundVolume(Volume);

通过编程来实现播放声音文件。文件地址是 C:\Program Files（x86）\Robomatter Inc\ROBOTC Development Environment 4. X\EV3 System Files\Sounds。

可以打开该文件，将要播放的声音文件放入，在程序中写入文件名，当下载程序时，就会自动将想要播放的声音文件下载到 EV3 主控器之中。

使用 playTone 语句以及变量知识编程，来实现一个音调从低到高，而且变化速度越来越快的程序。

```
int i,s;
i =100;s =100;
while()
{
playTone(s,i);
}
```

对照下面的音频频率表，尝试编程制作一小段曲子，如图 2-22 所示。

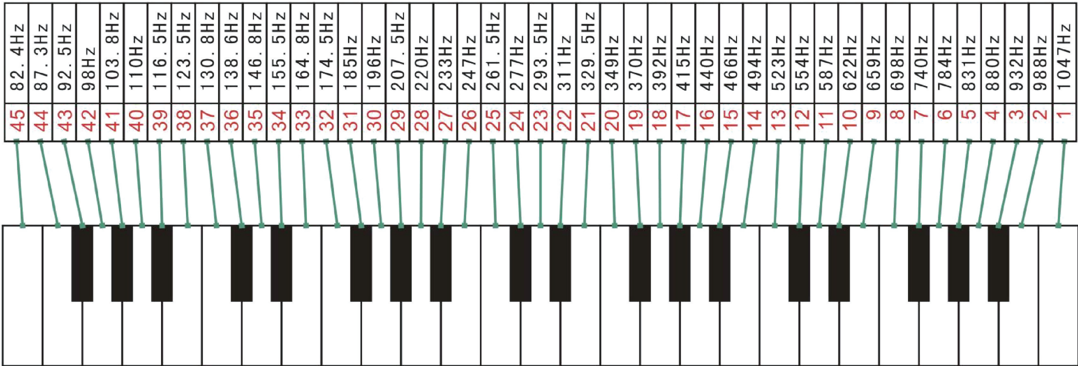


图　2-22

```
task main()
{
playTone(223,50);sleep(500);
playTone(247,50);sleep(500);
playTone(261.5,50);sleep(500);
playTone(277,50);sleep(500);
playTone(293.5,50);sleep(500);
playTone(311,50);sleep(500);
playTone(329.5,50);sleep(500);
playTone(349,50);sleep(500);
playTone(370,50);sleep(500);
playTone(392,50);sleep(500);
playTone(415,50);sleep(500);
playTone(440,50);sleep(500);
playTone(466,50);sleep(500);
playTone(494,50);sleep(500);

}
```

这里的频率只是做一个例子，读者可以根据自已的需求进行修改。

2.12　函数

2.12.1　编程知识

函数是一系列 C 语句的集合，可以完成某个会重复使用的特定功能。需要该功能的时候，直接调用该函数即可，不必每次都重复编写一大堆相同或类似的代码。需要修改该功能的时候，也只要修改和维护这个函数即可。总之，将语句集合成函数，好处就是方便代码重用，并且一个好的函数名，可以让人一眼就知道这个函数实现的是什么功能，方便维护。

定义前进函数(使用 void 定义此函数)。

```
    void Forward( )
{
    setMotorSpeed(leftMotor,50);
    setMotorSpeed(rightMotor,50);
    sleep(1000);
}
task main()
    {
    Forward();
    }
```

这里的 Rorward( ) 直接就可以代表小车以 50 的速度运行 1s。

定义可改变前进速度和时间的函数。

```
void Forward(int a,int b)
{
  setMotorSpeed(leftMotor,a);
  setMotorSpeed(rightMotor,a);
  sleep(b);
task main()
  {
  Forward(30,  1000);
  }
}
```

int a 的数值，意为双电动机以 30 的速度前进。

int b 的数值，意为运行 1s。

2.12.2 编程实例

（1）使用函数控制小车，完成虚拟世界迷宫挑战。

```
void allmotor(a,b)
{setMotorSpeed(motorA,a);
  setMotorSpeed(motorB,b);
}
task main()
{
  allmotor(100,100);
  sleep(5000);
  allmotor(-100,100);
  sleep(1000);
  allmotor(100,-100);
  sleep(1000);
  allmotor(100,100);
  sleep(5000);
}
```

（2）使用函数编程完成在黑圈内或桌面上的巡逻任务，不得出界或掉落。

```
void allmotor(int a,int b)
{setMotorSpeed(motorB,a);
  setMotorSpeed(motorC,b);
}

task main()
{
  while(1)
    if(getColorReflected(S1)<50)
      {resetGyro(S2);
      while(getGyroDegrees(S2)>-90)
        {allmotor(-100,100);
        }
      }
  else
  {allmotor(100,100);
  }

}
```

当颜色传感器检测到灰度值小于 50（压到黑线），重置陀螺仪，小车开始转弯，直到陀螺仪角度小于−90°时，小车又开始前进（小车转 90°）。

2.13 switch...case

2.13.1 编程知识

```
switch...case(break)
switch (表达式)
{
case 值 1 :语句 1 break;
case 值 2 : 语句 2 break;
...
default :...;
}
```

从表达式值等于某个 case 语句后的值开始，它下方的所有语句都会一直运行，直到遇到一个 break 为止。随后，switch 语句将结束，程序从 switch 结束大括号之后的第一个语句继续执行，并忽略其他 case，如图 2-23 所示。

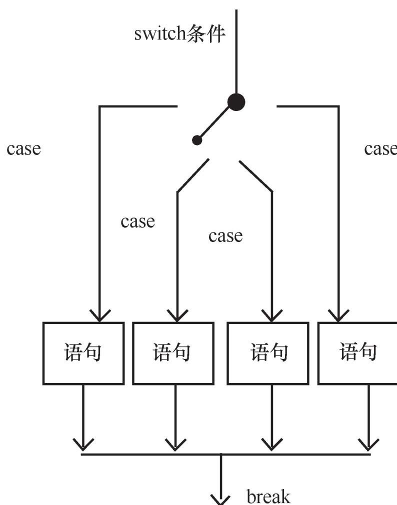


图 2-23

假如任何一个 case 语句的值都不等于表达式的值，那么就会运行 default 之下的语句。

假如表达式的值和任何一个 case 标签都不匹配，同时没有发现一个 default 标签，程序会跳过整个 switch 语句，从它的结束大括号之后的第一个语句继续执行。

该语句可以针对多种情况做出不同的应对。

根据触动传感器按压的次数，在屏幕上显示英文字母 A，B，C，D；

```
switch(a)
{
case 1:setLEDColor();break;
case 2:setLEDColor(); break;
...
default:...;
}
```

2.13.2 编程实例

（1）编程控制触动传感器，不按它的时候不亮，按一次亮橘色，按两次亮红色，按三次亮绿色，按四次后又从头开始，如下：

```
task main()
{
  int a=0;
  while(1)
  {
    if(getBumpedValue(E)==1)
    {a++;
      sleep(500);
    }
     switch(a)
     {
       case 0:setLEDColor(ledOff);break;
       case 1:setLEDColor(ledOrange);break;
       case 2:setLEDColor(ledRed);break;
       case 3:setLEDColor(ledGreen);break;
       default:a=0;
     }
}
}
```

（2）通过编程来实现让一个变量作为月份，每过 1s 把月份加大，直到 12 月后重新回到 1 月。根据月份不同在屏幕上显示 spring、summer、autumn、winter 四季名称（使用 switch case 语句）。

```
int a=1;
task main()
{
  while(1)
  {displayTextLine(1,"%d",a);
    displayTextLine(2,"winter");
  switch(a)
  {
    case 1:displayTextLine(2,"winter");break;
    case 2:displayTextLine(2,"winter");break;
    case 3:displayTextLine(2,"spring");break;
    case 4:displayTextLine(2,"spring");break;
    case 5:displayTextLine(2,"spring");break;
    case 6:displayTextLine(2,"summer");break;
    case 7:displayTextLine(2,"summer");break;
    case 8:displayTextLine(2,"summer");break;
    case 9:displayTextLine(2,"autumn");break;
    case 10:displayTextLine(2,"autumn");break;
    case 11:displayTextLine(2,"autumn");break;
    case 12:displayTextLine(2,"winter");break;
    default:a=0;
  }
sleep(1000);
a++;
  }
}
```

2.14　线程

2.14.1　编程知识

线程是程序中的一个执行流，每个线程都有自己的专有寄存器，但代码区是共享的，即不同的线程可以执行同样的函数。

多线程：是指程序中包含多个执行流，即在一个程序中可以同时运行多个不同的线程来执行不同的任务，也就是说允许单个程序创建多个并行执行的线程来完成各自的任务。

C 语言最初并未设计多线程的机制，随着软、硬件的发展及需求的发展，后来 C 语言才开发了线程库，以支持多线程的操作、应用。

程序在进入循环或者在 sleep( )中，无法灵活改变其中的参数，那么如何在互不干扰的情况下实现多个任务控制呢？这里需要用到线程知识。

使用之前所学的语句实现两个触动传感器对两个大型电动机分别进行控制，按下一侧传感器，则一侧电动机开始旋转。

```
task one()// 线程一
{
}
task main()// 主线程
{
startTask(one);// 开始线程一
stopTask(one);// 停止线程一
}
```

2.14.2　编程实例

使用线程知识实现两个触动传感器对两个电动机的控制，要求每次按下一侧触动传感器，则对应的电动

机转一圈，另一侧也一样，且相互不影响。

```
task one()
{while(1)
  {
  if(getTouchValue(E)==1)
    {
    resetMotorEncoder(motorA);
    setMotorTarget(motorA,360,100);
    waitUntilMotorStop(motorA);
    }
  }
}
task main()
{
  startTask(one);
while(1)
  {
  if(getTouchValue(G)==1)
    {
    resetMotorEncoder(motorD);
    setMotorTarget(motorD,360,100);
    waitUntilMotorStop(motorD);
    }
  }

}
```

2.15 随机数

2.15.1 编程知识

计算机用数学方法产生随机数，是根据确定的算法推算出来的，因此严格来说，用数学方法在计算机上产生的“随机数”不能算是真正的随机数，一般称为“伪随机数”。不过这些伪随机数，只要符合一些统计要求，如均匀性、随机性、独立性等，就可以作为真正的随机数来使用。rand()%N 为随机数函数，N 为正整数。

例如：

```
displayTextLine(2,"%d",rand()%11);
```

"%" 后写 11 的意思为取 -10 到 10 之间的随机整数。

试显示 4 到 10 之间的随机整数。取得方法：首先取两个数字之间整数的个数，然后通过数学运算转换成所需的随机数。

例如：

```
rand()%2 +7
```

这里 rand()%4 意为取 -3 到 3 之间的随机整数，需要的是 4 到 10 之间的随机整数，而 4 减去 -3 得 7，10 减去 3 得 7，所以 ran()%4 +7 即为 4 到 10 之间的随机整数。

2.15.2 编程实例

（1）尝试使用陀螺仪以及随机函数，让一辆双电动机小车随机移动（每隔 1s 就随机选择一个方向）。

```
int x=0;
task main()
{while(1)
{
  x=rand()%360; //x数值为-359到359之间取随机数
  while(getMotorEncoder(motorB)<720) //当电动机旋转两圈之后跳出循环
  {
    setMotorSpeed(motorB,50-(getGyroDegrees(S2)-x)*2);
    setMotorSpeed(motorC,50+(getGyroDegrees(S2)-x)*2);
  }
  resetMotorEncoder(motorB);
}

}
```

（2）如果使用 switch...case，怎样让一辆双电动机小车随机移动呢（每隔 1s 就随机选择一个方向）？代码如下所示。

```
void mov(int x)
{
  while(getMotorEncoder(motorB)<360)
  {
    setMotorSpeed(motorB,50-(getGyroDegrees(S2)-x)*2);
    setMotorSpeed(motorC,50+(getGyroDegrees(S2)-x)*2);
  }
}
int a=0;
task main()
{
  while(1)
  {
    a=rand()%3+3;
    switch(a)
    {
    case 1:mov(0);break;
    case 2:mov(-40);break;
    case 3:mov(50);break;
    case 4:mov(-120);break;
    default:mov(160);break;
    }
    resetMotorEncoder(motorB);
  }
}
```



乐高机器人设计及搭建绝妙技法

定价：49.80元

ISBN 978-7-111-54516-3



乐高EV3机器人创意编程精彩实例

定价：89.00元

ISBN 978-7-111-55497-4



VEX IQ创意编程与精彩实例

定价：69.00元

ISBN 978-7-111-56111-8

ROBOTC for LEGO EV3 基础编程与实例

特长生的培养手册 国际大赛的入场券

码高机器人教育 编著

地址：北京市百万庄大街22号
邮政编码：100037
电话服务
服务咨询热线：010-88361066
读者购书热线：010-68326294
010-88379203
网络服务
机工官网：www.cmpbook.com
机工官博：weibo.com/cmp1952
金书网：www.golden-book.com
教育服务网：www.cmpedu.com
封面无防伪标均为盗版


机械工业出版社
微信公众号


机工互联网家


码高机器人

欲获取更多资源及图书信息请关注

机械工业出版社计算机分社
微信二维码—IT有得聊

上架指导 科普玩具/EV3
ISBN 978-7-111-60056-5
策划编辑◎杨源 / 封面设计◎史淑娴



定价：49.80元